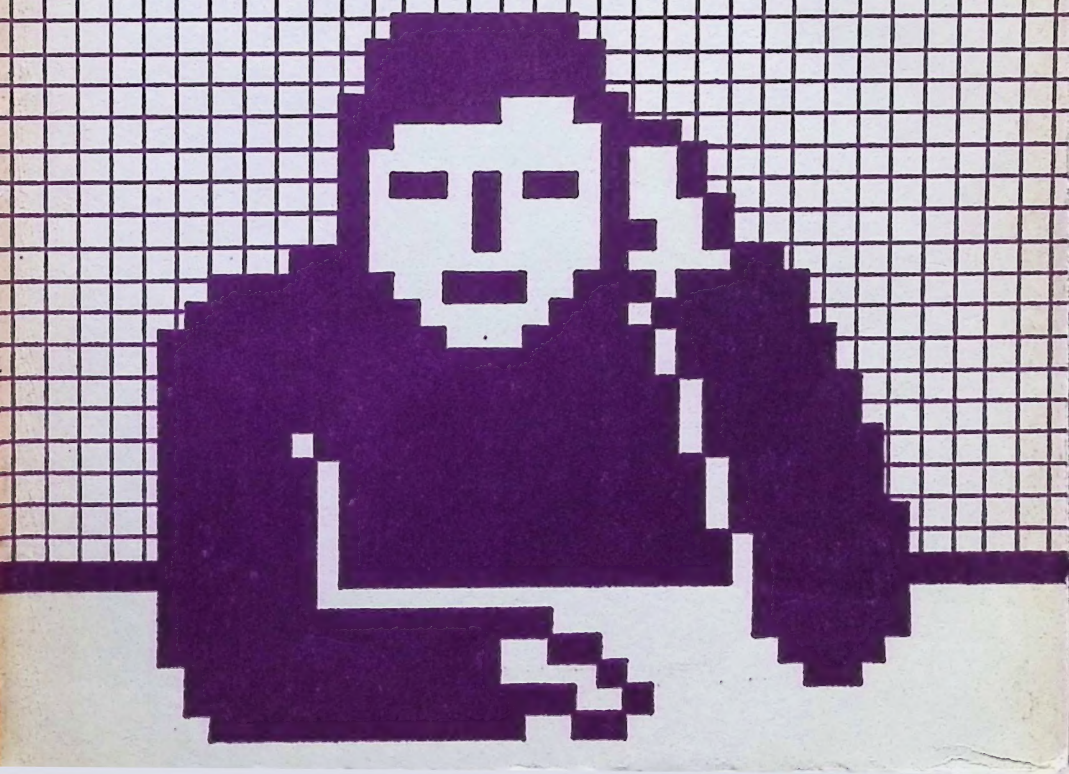
Mikrokomputery

Jacques Boisgontier

Sophie Brébion

BASIC

dla wszystkich



Basic
dla wszystkich

Mikrokomputery

Komitet Redakcyjny

Sekretarz WOJCIECH CELLARY
ZUZANNA GRZEJSZCZAK
ANDRZEJ KOBUS
Przewodniczący ROMUALD MARCZYŃSKI
PIOTR MISIUREWICZ
WOJCIECH NOWAKOWSKI
MACIEJ STOLARSKI
HALINA TEMPCZYK
JÓZEF WINKOWSKI
JAN ZABRODZKI

Jacques Boisgontier Sophie Brébion

Basic dla wszystkich

Z języka francuskiego przełożyła

ANNA ISZKOWSKA

Wydawnictwa Naukowo-Techniczne
Warszawa 1987

Dane o oryginale
Basic pour tous
Initiation + Programmes
par
Jacques Boisgontier et *Sophie Brébion*
© Editions du P.S.I. – 77402 Lagny Cedex, 1984

Opiniodawca *Jan Bielecki*
Redaktor *Maria Frankowska*
Redaktor techniczny *Barbara Chojnacka-Flisiuk*
Okładkę i strony tytułowe projektował *Juliusz Rybicki*

681.3.06

Książka uczy programowania w języku Basic. Czytelnik bardzo szybko zaznajamia się z elementarnymi instrukcjami oraz podstawowymi pojęciami programowania, jak zmienne, program, pętla, podprogram...
Dzięki licznym przykładom programów bezpośrednio komentowanych w tekście, Czytelnik bez trudu nabywa niezbędnego doświadczenia umożliwiającego pisanie własnych programów z dziedziny zarządzania nauczania, gier itp.
Książka jest przeznaczona dla Czytelników pragnących szybko i dobrze opanować podstawy programowania w języku Basic.

© Copyright for the Polish edition
by Wydawnictwa Naukowo-Techniczne
Warszawa 1987

All rights reserved
Printed in Poland

ISBN 83-204-0858-X

Spis treści

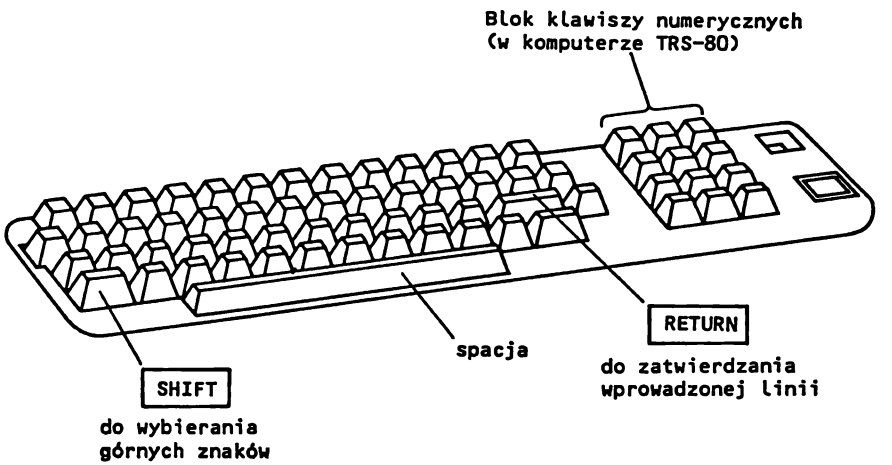
Jak wygląda twój komputer?/7
Tryb bezpośredni (zwany także MODE DIRECT)/11
Zapamiętywanie wartości: zmienne/14
Zapamiętywanie instrukcji: program/21
Wprowadzanie danych w trakcie wykonywania programu:
INPUT "tekst"; zmienna/25
Pętla: GOTO nr linii/28
Interludium: w jaki sposób przechować program?/31
Decyzje: IF warunek THEN instrukcja (JEŻELI warunek TO instrukcja)/33
Pętla automatyczna: FOR... NEXT/39
Teksty/48
Uruchamianie programów: BREAK-CONT-STOP/60
Podprogramy: GOSUB/RETURN/64
Interludium: w sprawie schematów blokowych/67
DATA/READ/RESTORE/68
Tablice/75
Wyprowadzanie wyników/94
Bezpośrednie adresowanie ekranu/104
Przechwytywanie znaku z klawiatury/107
Liczby losowe/111
Grafika .../117
... w komputerze TRS-80/118
... w komputrze APPLE II – niski poziom/131
... w komputerze APPLE II – wysoki poziom/139
Dodatek I: Edytor/160
Dodatek II: Zestawienie instrukcji języka BASIC/162
Dodatek III: Sygnalizacja błędów/170
Dodatek IV: Tablica kodu ASCII/175

Jak wygląda twój komputer?

Postanowiłeś zapoznać się z językiem Basic. Przed sobą masz komputer, a w ręku książkę.
Jak wygląda ten twój nowy towarzysz, którego zamierzasz "oswoić"?

Ekran

Może to być ekran zintegrowany z komputerem (TRS-80, APPLE II) albo ekran telewizyjny, a więc niezależny od komputera. To właśnie za pośrednictwem ekranu komputer będzie odpowiadać na twoje pytania.
W jaki sposób można jednak zadawać pytania komputerowi?
Jak dotąd, komputery nie rozumieją ludzkiej mowy, musisz więc skorzystać z pośrednictwa klawiatury.



Klawiatura

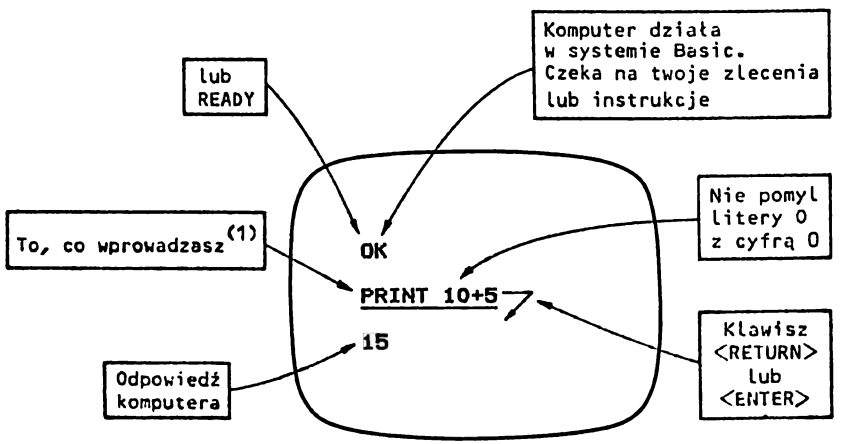
Jak widzisz, przypomina ona klawiaturę maszyny do pisania. Jest tutaj wszystko – litery, cyfry, znaki interpunkcyjne, symbole matematyczne, klawisz spacji. Jest to najczęściej klawiatura typu QWERTY, co odpowiada układowi liter w drugim od góry rzędzie klawiszy. Niekiedy masz także do dyspozycji oddzielny blok klawiszy numerycznych. Okaże się on bardzo przydatny przy wprowadzaniu danych numerycznych.
To właśnie dzięki klawiaturze będziesz się mógł porozumiewać, czyli prowadzić **dialog**, z komputerem.

W celu oswojenia się z tym nowym sposobem porozumiewania się naciśnij bez obawy (lecz delikatnie!) poszczególne klawisze. Niczym nie ryzykujesz... komputer także nie.

Podczas zapoznawania się z klawiaturą natknąłeś się na pewne klawisze, których znaczenie nie jest ci jeszcze znane.

RETURN: (oznaczany niekiedy jako ENTER)

Za pomocą tego klawisza "zatwierdzasz" informację, którą chcesz przesłać do komputera. Dopóki go nie naciśniesz, możesz kasować wprowadzane znaki klawiszem ←.



(1) Na rysunkach przedstawiających ekran wyróżniamy przez podkreślenie informacje wprowadzane przez użytkownika, odróżniając je w ten sposób od odpowiedzi wysyłanych przez komputer.

Naciskanie klawisza <**RETURN**> *w celu zatwierdzenia wprowadzanej instrukcji lub zlecenia powinno stać się odruchem.*

Jeżeli o tym zapomnisz, komputer pozostanie głuchy... na wszystkie twoje złorzeczenia!

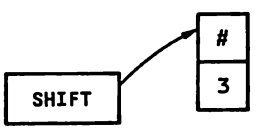
Uwaga: Komunikat **OK** lub **READY** (w zależności od typu komputera) sygnalizuje, że znajdujesz się w systemie Basic.

←:

Klawisz ten umożliwia, dopóki nie naciśniesz klawisza <RETURN>, skasowanie ostatniego znaku.
Kursor cofa się kasując ostatni znak, znak poprzedni itd., w miarę jak przesuwa się w lewo.
Jak widzisz. klawisz ← jest bardzo użyteczny. Będziesz go używał do poprawiania błędów, o ile oczywiście dostrzeżesz je **zanim** zatwierdzisz całą linię klawiszem?... <RETURN>.
Jeżeli nie – cierpliwości, wyjaśnimy nieco później, w jaki sposób można zmienić utworzoną już linię (patrz dodatek I).
W celu wprowadzenia znaku #.

SHIFT

Korzystasz z tego klawisza przy wprowadzaniu górnych znaków klawiszy dwuznakowych.
Na przykład:
W celu wprowadzenia znaku #:



Uwaga. Klawisz SHIFT nie jest stabilny. Musisz **najpierw** nacisnąć klawisz SHIFT a następnie, **nie zwalniając go**, nacisnąć klawisz z żądanym znakiem.

Podsumowanie

- **Klawiatura** umożliwia prowadzenie **dialogu** z komputerem.
- **Ekran** umożliwia **zobrazowanie** "konwersacji", przebiegającej między tobą a maszyną.
- Klawisz **RETURN zatwierdza** informację przesyłaną do komputera.
- Klawisz ← **cofa** kursor z jednoczesnym **kasowaniem** znaków napotykanych na jego drodze.
- Klawisz **SHIFT** jest używany przy wprowadzaniu **górnego znaku** klawisza dwuznakowego.

Tryb bezpośredni (zwany także MODE DIRECT)

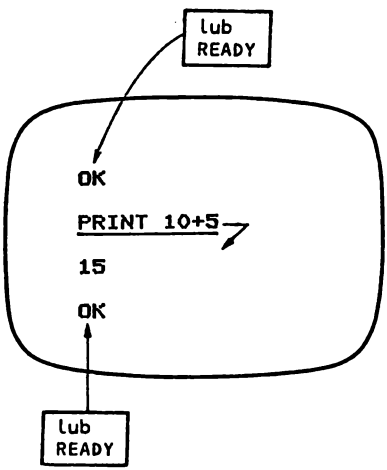
Aby prowadzić dialog z komputerem nie wystarczy niestety znajomość jego klawiatury, trzeba także mówić jego językiem, językiem Basic. Nie niepokój się jednak; zobaczysz jakie to proste!
Opanujesz biegle ten nowy język już po krótkiej praktyce.

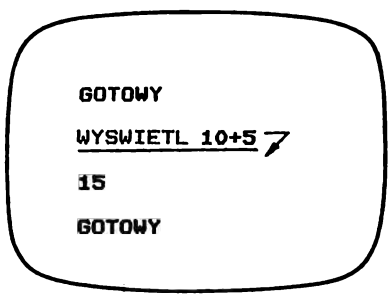
Aby niezwłocznie rozpocząć konwersację, wprowadź jakąś instrukcję, pamiętając o zatwierdzeniu jej klawiszem <RETURN>. Komputer odpowie ci natychmiast.

Zaczynamy:

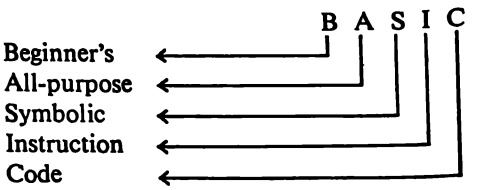
W języku angielskim
- *Ty*: PRINT 10+5
- *On*:15

W tłumaczeniu na język polski
- *Ty*: WYŚWIETL 10+5
- *On*:15

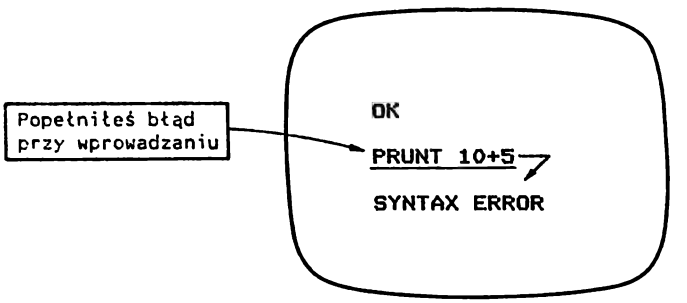




Wyjaśnimy, co oznacza nazwa
(dla początkujących)
(o wszechstronnym zastosowaniu)

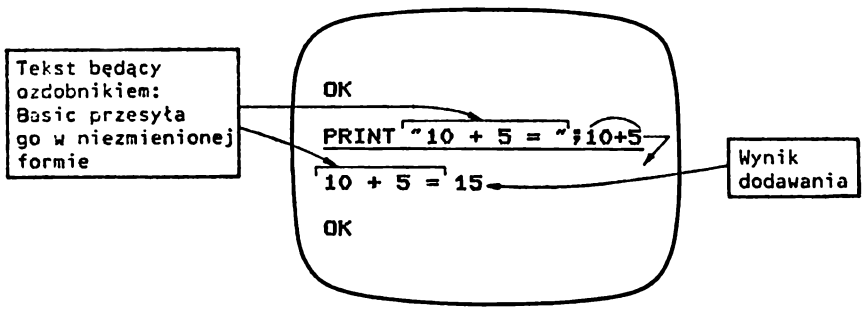


Język Basic, tak jak wszystkie inne języki, ma składnię, której musisz przestrzegać. W przeciwnym przypadku komputer nie zrozumie przesyłanej przez ciebie informacji. Zasygnalizuje to komunikatem SYNTAX ERROR (błąd składniowy), po wyświetleniu którego będziesz oczekiwać na ponowne wprowadzenie bezbłędnej instrukcji.



Załóżmy, że dostrzegłeś ten niewybaczalny (mówimy w imieniu maszyny!) błąd przed naciśnięciem klawisza <RETURN>.
Co mógłbyś zrobić, aby naprawić to przeoczenie?

Jak już zapewne zauważyłeś, komputer ogranicza się do dawania odpowiedzi pozbawionej jakiegokolwiek komentarza.
Musisz sam zadbać o wzbogacenie jej formy.
W tym celu wystarczy wprowadzić **tekst**, umieszczony między **znakami cudzysłowu.**

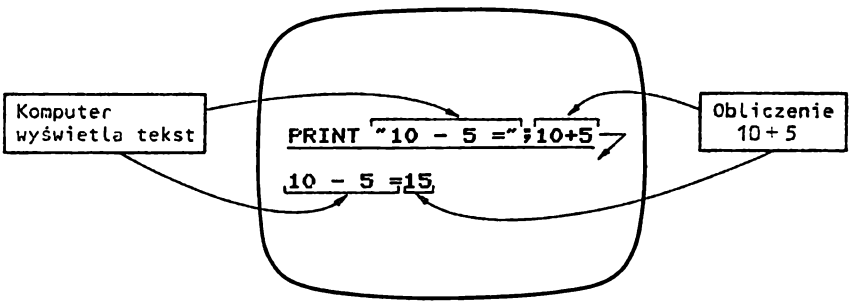


Uwaga na znaki cudzysłowu!

To właśnie znaki cudzysłowu sygnalizują komputerowi, że chodzi tutaj o tekst. Tekst ten jest całkowicie dowolny. Mógłbyś równie dobrze napisać:

```
PRINT "10 - 5 =";10+5
```

Komputer wydzieli tekst ograniczony znakami cudzysłowu i wyświetli go w niezmienionej postaci.



Świadczy to o jego dokładności, ale też jednocześnie o braku "inteligencji".

Podsumowanie

- Tryb *bezpośredni* (**DIRECT**) umożliwia "natychmiastowe" poznanie wyniku działania instrukcji.
- Instrukcja **PRINT** (**WYŚWIETL**) jest używana do wyświetlania informacji na ekranie.
- Komunikat **SYNTAX ERROR** oznacza, ze komputer nie zrozumiał twojego polecenia. Prosi cię o ponowne, tym razem bezbłędne, wprowadzenie instrukcji.
- **Znaki cudzysłowu** "..." wskazują maszynie, że musi ona jedynie skopiować ograniczony nimi tekst.

Zapamiętywanie wartości: zmienne

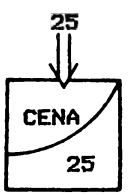
Instrukcja PRINT, jakkolwiek bardzo istotna, nie ujawnia prawie zupełnie możliwości maszyny.

Instrukcja LET zmienna = wartość

Instrukcja ta umożliwia przechowanie w "pamięci" wartości, które będą potem używane (na przykład do prowadzenia obliczeń).
Zmienną możemy sobie wyobrażać jako komórkę (w ten zresztą sposób będziemy ją symbolicznie oznaczać), we wnętrzu której umieszczamy jakąś wartość. Komórka ta jest *indentyfikowana* przez nadaną jej **nazwę**.
Aby *przyporządkować* (nadać) zmiennej o nazwie 'CENA' wartość 25, piszemy:

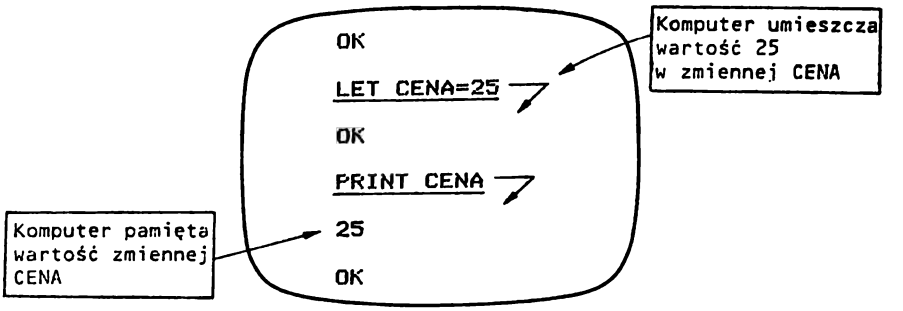
```
LET CENA=25
```

Basic umieszcza wartość 25 w komórce CENA.

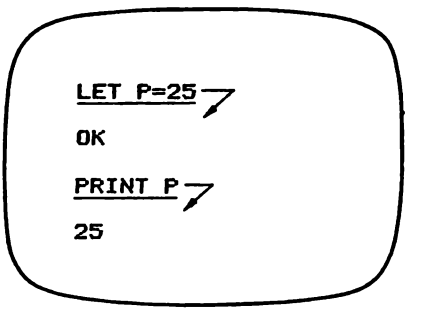



```
LET CENA=25
```

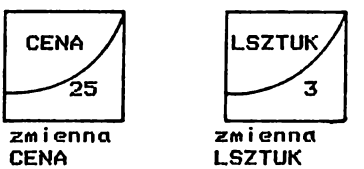
Komputer pamięta wartość zmiennej CENA. Aby wyświetlić tę wartość, wystarczy napisać PRINT CENA.

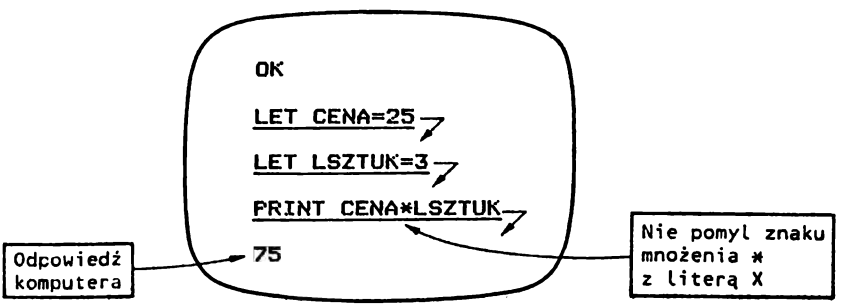


Nazwa nadawana zmiennej, w tym przypadku 'CENA', jest całkowicie dowolna. Aby to sprawdzić, napisz:

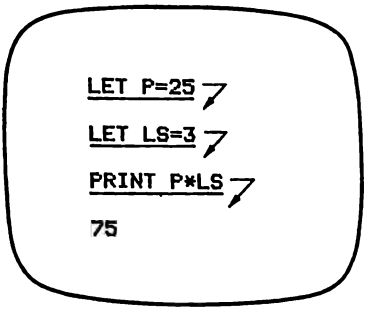


Ważne jest jednak, aby nazwa wybierana dla oznaczenia zmiennej była możliwie jak najbardziej znacząca.
Obliczmy teraz łączną wartość trzech produktów w cenie 25 zł. W tym celu użyjemy dwóch zmiennych (dwóch komórek).





Nazwy zmiennych 'CENA' i 'LSZTUK' można zastąpić nazwami 'P' i 'LS'.



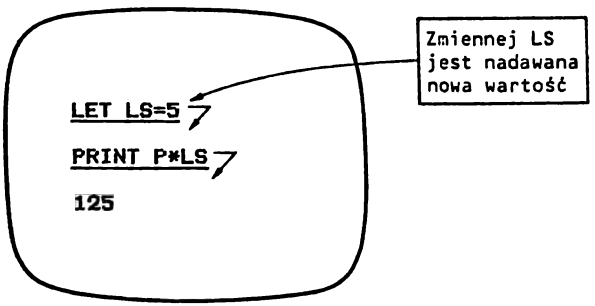
Otrzymamy ten sam wynik, pisząc:

```
LET P=25
LET LS=3
LET S=P*LS
PRINT S

75
```

Zobaczmy teraz, w jaki sposób zmieniamy wartość zmiennej LS. Aby nadać jej wartość 5, wystarczy napisać:

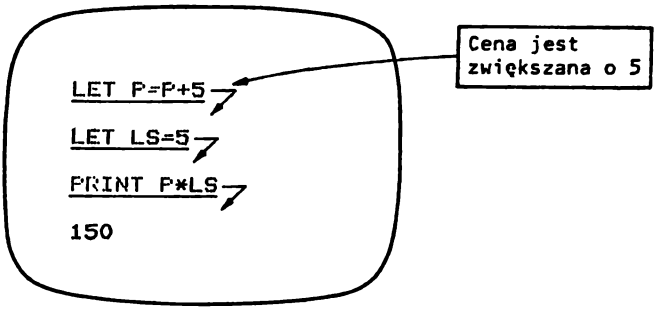
```
LET LS=5
```

(zmienna P zachowuje swą dawną wartość)



Komputer "zapomina" oczywiście o poprzedniej wartości (3) zmiennej LS.

Załóżmy teraz, że cena wzrasta o 5 złotych. Moglibyśmy oczywiście napisać: LET P = 30.
Możemy jednak uzyskać ten sam wynik w inny sposób:

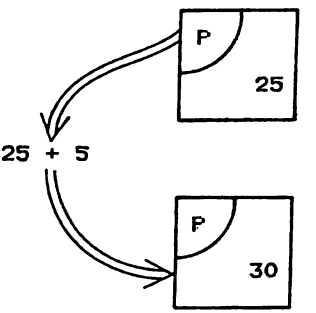


LET P = P + 5 nie powinno być rozumiane, jako równanie algebraiczne. Komputer interpretuje ten napis w następujący sposób:

① *Pobiera zawartość komórki P*

② *Dodaje do niej 5*

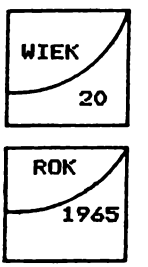
③ *Umieszcza sumę w komórce P*



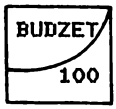
Przykłady

Wyznaczmy datę urodzenia na podstawie podanego wieku:

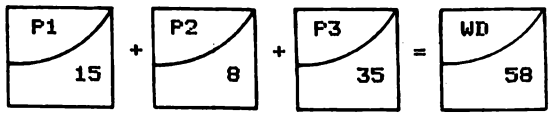
```
LET WIEK=20
LET ROK=1985-WIEK
PRINT "Data urodzenia: ";ROK
Da a ur dzenia: 1965
```



Obliczmy teraz sumę wydatków (P1, P2, P3) oraz pozostałą kwotę:




```
LET BUDZET=100
LET P1=15
LET P2=8
LET P3=35
LET  WD=P1+P2+P3
```



Suma wydatków

```
PRINT "Wydatki razem: ";WD
Wydatki razem: 58
LET RESZTA=BUDZET-WD
PRINT "Pozostalo:";RESZTA
Pozostalo: 42
```

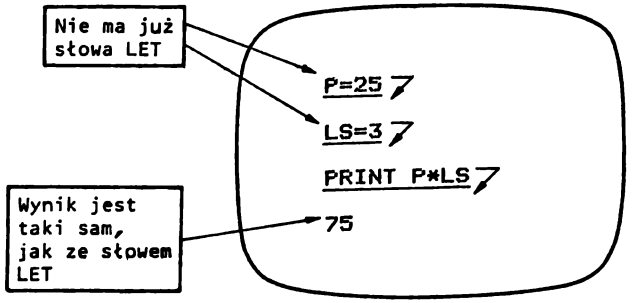
Aby sprawdzić wartości zmiennych P1, P2 i P3, napiszmy:

```
PRINT P1,P2,P3
15          8          35
```

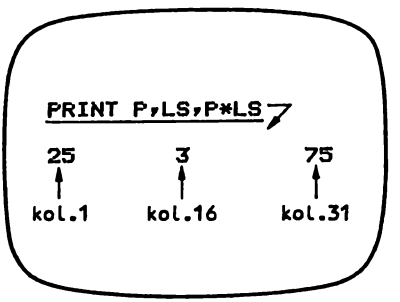
Jeżeli popełniliśmy błąd przy ustalaniu ceny P2, wystarczy napisać:

```
LET P2=18
LET WD=P1+P2+P3
PRINT "Wydatki razem: ";WD
Wydatki razem: 68
```

Chcąc przydzielić zmiennej jakąś wartość, używaliśmy instrukcji LET. Większość istniejących obecnie wersji języka Basic dopuszcza pomijanie słowa LET.



Aby wyświetlić w jednej linii cenę **P**, *liczbę sztuk* **LS** *i łączną wartość* **P*LS**, *rozdzielamy zmienne* **przecinkami.**



Wyniki są wyświetlane w kolumnach 1, 16 i 31 (co 15 pozycji).

Średnik *blokuje wprawdzie przeskok do nowej linii, ale nie powoduje tabulacji wyników. Są one wyprowadzane "ciurkiem", rozdzielone jedynie pojedynczymi spacjami.*[1].

```
PRINT P;LS;P*LS

25  3  75
```

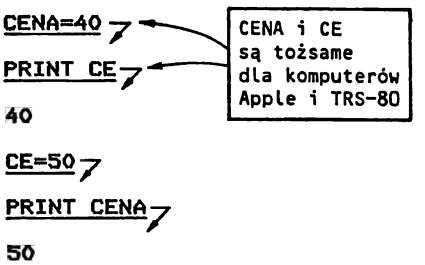
[1] Spacje rozdzielające wyprowadzane liczby są związane z formatem wyświetlania liczby w języku Basic. Liczba jest wyświetlana jako ciąg cyfr (ewentualnie z kropką dziesiętną) zakończony spacją. Liczbę ujemną poprzedza znak minus, natomiast znak liczby dodatniej, w zależności od wersji języka, jest pomijany (APPLE II) lub zastępowany spacją (przyp. tłum.).

Uwagi dotyczące nazw zmiennych:

Pierwszy znak musi być literą. Pozostałe znaki mogą być zarówno literami, jak i cyframi.

A2 jest poprawne
2A jest błędne

W wielu komputerach, jak np. TRS-80 i APPPLE II, są brane pod uwagę tylko dwa pierwsze znaki:



Dla systemu Basic, CENA i CE są jedną i ta samą zmienną.

Uwaga! W większości komputerów nie można używać nazw zmiennych zawierających słowa kluczowe języka Basic. Na przykład TOTAL jest nazwą zabronioną, ponieważ TO jest słowem kluczowym języka Basic. TTAL jest natomiast nazwą dopuszczalną.

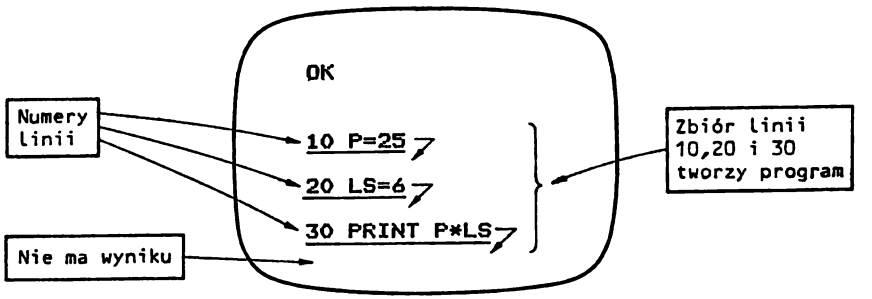
Podsumowanie

- **Zmienna** jest "komórką", identyfikowaną przez nazwę, w której można umieścić dowolną wartość.
- Instrukcja **LET** przydziela zmiennej wartość.
- Do zmiennej X można dodać jakąś wartość, pisząc:
  LET X = X + wartość (lub X = X + wartość).
- **Przecinek** umożliwia wyświetlenie wyników w kolumnach 1, 16, 31 ...
- **Średnik** nie powoduje tabulacji wyników.

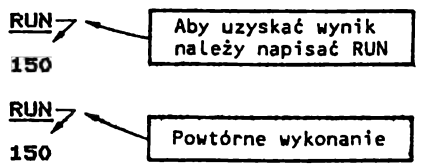
Zapamiętywanie instrukcji: program

Zamiast wielokrotnie wprowadzać ten sam ciąg instrukcji, możemy go umieścić w pamięci komputera.

W celu zapamiętania ciągu instrukcji, wprowadzamy instrukcje poprzedzając je **numerem linii** (10, 20, 30,...).
Taki ciąg instrukcji nazywamy **programem.**



Stwierdzamy, że nie uzyskaliśmy wyniku: *Aby otrzymać wynik, należy zażądać* **wykonania programu** *wydając zlecenie* **RUN.**



Basic wykonuje instrukcje kolejno według wzrastających numerów. Program może być wykonywany wielokrotnie.

Uwaga: Kolejność wprowadzania linii jest dowolna, tzn. nie musi być zgodna z numeracją.

Modyfikowanie linii

Najprostszą metodą jest ponowne wprowadzenie całej linii:

```
30 PRINT P,LS,P*LS↵

LIST↵

10 P=25
20 LS=6
30 PRINT P,LS,P*LS

RUN↵

25          6           150
```

Ponownie wprowadzono linię 30 (poprzednia została skasowana)

Wstawianie dodatkowej linii

Zaleca się numerowanie kolejnych instrukcji programu liczbami różniącymi się o 10. Aby wstawić nową linię, na przykład o numerze 25, wystarczy napisać:

```
25 PRINT "Cena:";P;"Sztuk:";LS↵

LIST↵

10 P=25
20 LS=6
25 PRINT "Cena:";P;"Sztuk:";LS
30 PRINT P,LS,P*LS

RUN↵

Cena:25 Sztuk:6
25                6                 150
```

Wstawienie linii 25

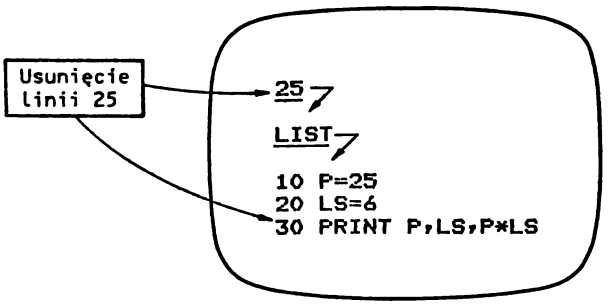
Wyświetlanie tekstu programu

Zlecenie LIST umożliwia wyświetlenie tekstu programu (lub części programu) umieszczonego w pamięci, ilekroć tylko uznamy to za konieczne.

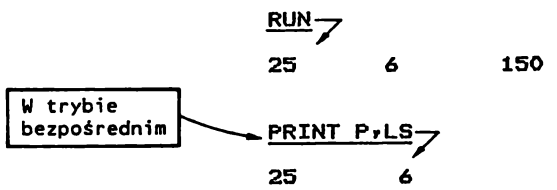
TRS-80 – MICROSOFT 5.:
LIST 10 wyświetla linię o numerze 10
LIST 10-30 wyświetla linie o numerach od 10 do 30
LIST 20- wyświetla linie, poczynając od linii o numerze 20 do końca programu
APPLE II
LIST 10, 30 wyświetla linie o numerach od 10 do 30

Usuwanie linii

Linię usuwa się pisząc jej numer i naciskając <RETURN>.



Po zakończeniu wykonywania programu (po zrealizowaniu zlecenia RUN), zmienne P i LS zachowują swoje wartości. Można się o tym przekonać pisząc w trybie bezpośrednim (to znaczy bez numeru linii): PRINT P, LS.



Chcąc się dowiedzieć, jak działa dana instrukcja, lepiej jest nieraz wprowadzić ją w trybie bezpośrednim niż poszukiwać wyjaśnienia w podręczniku.
Tryb bezpośredni jest również bardzo użyteczny podczas uruchamiania programu.
Z tego względu niezbędne jest nabycie umiejętności swobodnego korzystania z tego trybu.

Podsumowanie

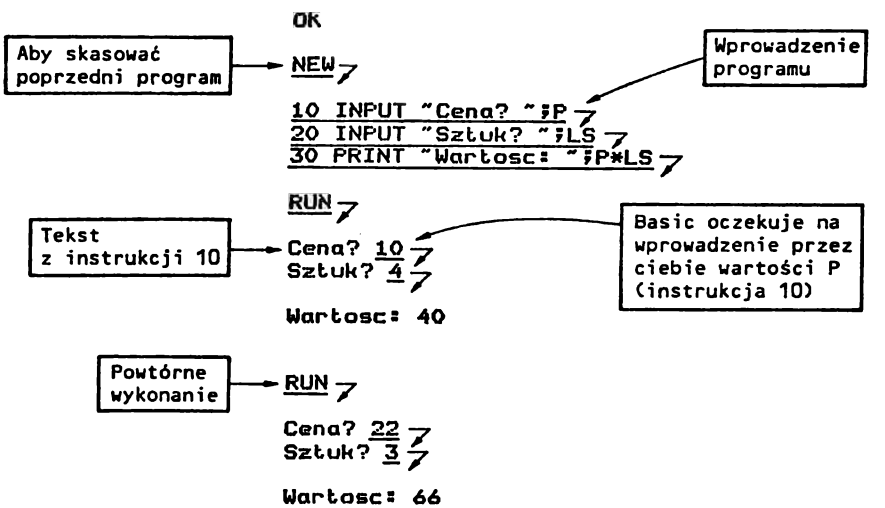
- **Program** jest ciągiem ponumerowanych instrukcji.
- Linie numeruje się zazwyczaj co 10, aby umożliwić wstawianie linii dodatkowych.
- Zlecenie **RUN** inicjuje wykonywanie programu.
- Zlecenie **LIST** wyświetla na ekranie program umieszczony w pamięci komputera.

Wprowadzanie danych w trakcie wykonywania programu: INPUT "tekst"; zmienna

Instrukcja **INPUT** *(WPROWADŹ) umożliwia dostarczenie danych* **w trakcie** *wykonywania programu,* **za pośrednictwem klawiatury.**

Basic, napotykając instrukcję INPUT podczas wykonywania programu, wyświetla na ekranie podany w tej instrukcji tekst (np. Cena?), a następnie **oczekuje na wprowadzenie** z klawiatury wartości zmiennej określonej w instrukcji INPUT (np. P).

Po wprowadzeniu przez operatora żądanej wartości i naciśnięciu <RETURN> wykonywanie programu jest kontynuowane od następnej linii.



Uwaga: Zlecenie **NEW** *umożliwia skasowanie starego programu umieszczonego w pamięci.* (W tym przykładzie zlecenie NEW nie było konieczne, ponieważ nowe linie zamazałyby i tak linie poprzedniego programu).

Instrukcja INPUT jest bardzo użyteczna, ponieważ umożliwia wykonywanie programu z różnymi danymi, bez konieczności modyfikowania samego programu.

Przykład

Obliczmy zużycie paliwa w samochodzie.

```
10 INPUT "Ile litrow? ";L
20 INPUT "Ile km?     ";KM
30 :
40 ZP=(L/KM)*100
50 :
60 PRINT "Zuzycie paliwa na 100 km: ";ZP
70 :
80 GOTO 10

RUN↵

Ile litrow? 40↵
Ile km?     520↵

Zuzycie paliwa na 100 km: 7.6
```

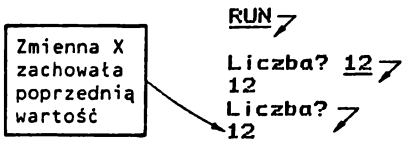
Jeżeli w odpowiedzi na instrukcję INPUT operator naciśnie jedynie klawisz <RETURN> nie podając żadnej wartości, to w przeważającej liczbie wersji systemu Basic zmienna określona w INPUT przyjmie wartość 0.

```
NEW↵

10 INPUT "Liczba? ";X
20 PRINT X
30 INPUT "Liczba? ";X
40 PRINT X

RUN↵

Liczba? 12↵
12
Liczba?↵
0
```

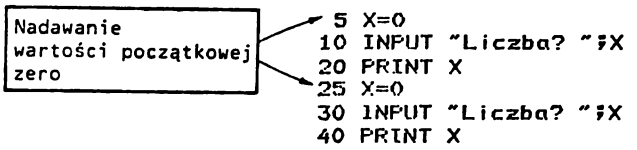
X ma wartość zero

Naciskasz jedynie <RETURN>

W komputerze TRS-80 zmienna zachowuje natomiast swą **dawną wartość.**

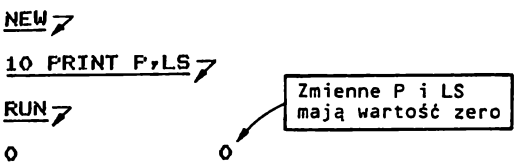


Aby temu zaradzić, należy napisać:



System Basic komputera APPLE II wymaga, aby operator wprowadził jakąś wartość przed naciśnięciem <RETURN>.

RUN *zeruje wszystkie zmienne programu.*



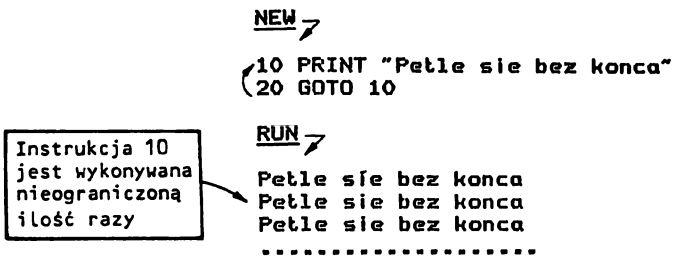
Podsumowanie

- Instrukcja **INPUT** umożliwia wprowadzanie wartości zmiennych **w trakcie** wykonywania programu.
- Zlecenie **NEW** kasuje program umieszczony w pamięci.

Pętla: GOTO nr linii

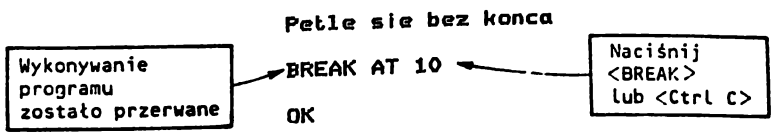
Instrukcja **'GOTO nr linii'** *umożliwia wielokrotne wykonanie tego samego ciągu instrukcji.*

Mechanizm działania GOTO przedstawimy na podstawie prostego przykładu (nie mającego poza tym żadnego praktycznego zastosowania):

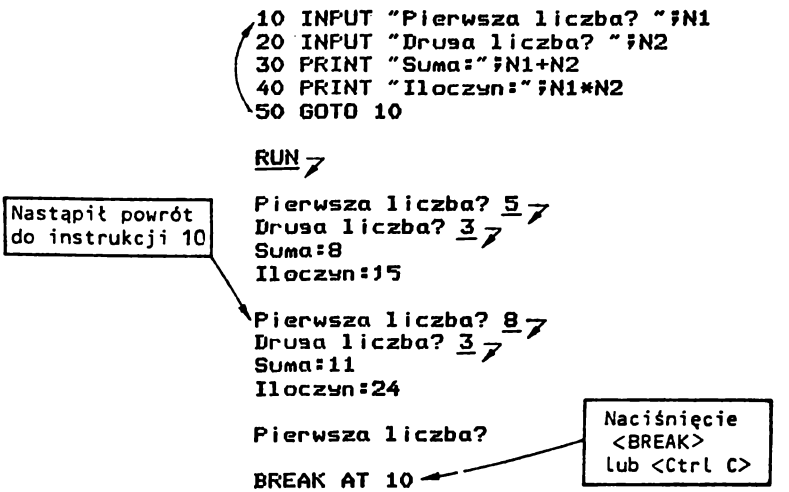


Basic realizuje instrukcję 10, a następnie przechodzi do wykonania instrukcji 20. Instrukcja '20 GOTO 10' oznacza 'PRZEJDŹ DO 10'. Tak więc instrukcja 10 będzie wykonywana ponownie.

Taki program nigdy się oczywiście nie zakończy... chyba że naciśniemy klawisz <BREAK> lub <Ctrl C>.



Użyjmy instrukcji GOTO do bardziej konkretnych celów. Przypuśćmy, że musimy wielokrotnie obliczać sumę i iloczyn dwóch liczb:.



Przykład

```
10 INPUT "Ile masz lat? ";WIEK
20 ROK=1985-WIEK
30 PRINT "Urodziles sie w roku: ";ROK
40 GOTO 10

RUN

Ile masz lat? 20
Urodziles sie w roku: 1965

Ile masz lat?
```

Aby spowodować przeskok o jedną linię przed wyświetleniem tekstu "Urodziłeś się w roku:", dołącz do programu instrukcję:

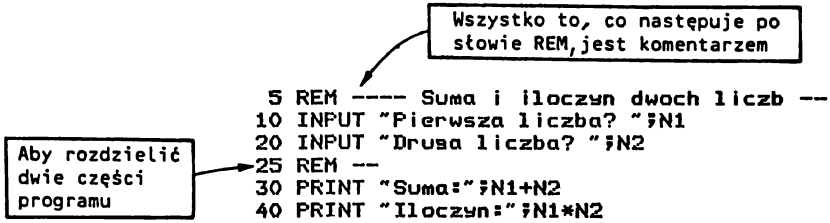
```
25 PRINT
```

Powoduje przeskok o jedną linię

Instrukcja REM

Nie jest to instrukcja w dosłownym znaczeniu tego słowa, ponieważ nie wpływa ona na przebieg programu.

REM (od angielskiego REMark – uwaga, notatka) *umożliwia wprowadzenie do programu komentarzy zwiększających jego czytelność.*



Wykonując program, Basic ignoruje wszystko to, co występuje w linii po słowie REM. W Microsoft 5 i TRS-80 REM może być zastąpione znakiem apostrofu.

```
5' -------- Suma i iloczyn dwoch liczb -----
```

Podsumowanie

- Instrukcja **'GOTO nr linii'** (*PRZEJDŹ DO nr linii*) powoduje kontynuację wykonywania programu od określonej linii.
- **REM** umożliwia dołączanie do programu komentarzy i jest pomijane podczas wykonywania programu.
- W celu zwiększenia czytelności programu pożądane jest dołączanie do niego komentarzy za pomocą REM (lub ').
- **<BREAK>** lub **<Ctrl C>** umożliwiają **przerwanie** wykonywania programu.

Interludium

W jaki sposób przechować program?

SAVE "nazwa programu"

Program wprowadzony do pamięci komputera jest tracony z chwilą wyłączenia napięcia.
Aby go zabezpieczyć, należy napisać (jeżeli dysponujesz jednostką dyskową):

SAVE "TEST", gdzie 'TEST' jest przykładową nazwą nadawaną programowi.

LOAD "TEST" umożliwi ci później ponowne załadowanie programu do pamięci komputera.

```
NEW

10 PRINT "PLEPLE"
20 GOTO 10

SAVE "TS1"↵

NEW↵

LIST↵

LOAD "TS1"↵

LIST↵

10 PRINT "PLEPLE"
20 GOTO 10
```

W pamięci operacyjnej nie ma programu

Program jest znowu w pamięci

W systemie komputera APPLE II należy pisać:

SAVE TEST (bez znaków cudzysłowu)
LOAD TEST

Gdy program ma być nagrywany na kasecie SAVE i LOAD przyjmują postać **CSAVE i CLOAD.**

Decyzje: IF warunek THEN instrukcja (JEŻELI warunek TO instrukcja)

Instrukcja **IF... THEN...** *umożliwia sprawdzenie określonego warunku i wykonanie jednej lub wielu instrukcji, jeżeli warunek ten jest spełniony.*

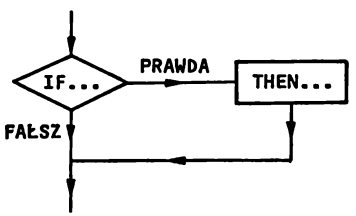
```
10 INPUT "Podaj liczbe? ";N1
20 IF (N1>10) THEN PRINT "Podales liczbe wieksza od 10"
30 Print "Kontynuacja"
40 GOTO 10

RUN↵

Podaj liczbe? 15↵
Podales liczbe wieksza od 10
Kontynuacja

Podaj liczbe? 8↵
Kontynuacja
```

Ta instrukcja jest wykonywana tylko wówczas, gdy N1 jest większa od 10

Instrukcja o numerze 20 oznacza:

20 JEŻELI N1 jest większe niż 10 TO wyświetl "Podałeś....."

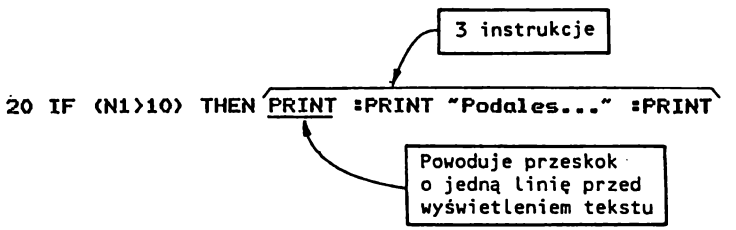
↑ IF ↑ THEN



Niezależnie od tego, czy występująca po słowie THEN instrukcja zostanie wykonana, czy też nie, program jest kontynuowany od instrukcji następującej po IF... THEN...

To właśnie ta instrukcja (jak również instrukcja GOTO) decyduje o sile komputera, gdyż daje ona możliwość wykonywania ciągów instrukcji w pewnych tylko przypadkach.

Uwaga: Aby wykonać warunkowo kilka instrukcji, wystarczy rozdzielić je znakiem : (dwukropek) [1].



Istnieją następujące operatory relacji:

= równe	> większe	= > równe lub większe
< mniejsze	< > różne	< = mniejsze lub równe

[1] Ogólnie – dwukropek, użyty jako separator instrukcji, umożliwia zapisanie w jednej linii programu wielu instrukcji języka Basic.
Wszystkie instrukcje występujące w linii po słowie THEN są wykonywane warunkowo. Jeśli bezpośrednio po słowie THEN występuje instukcja 'GOTO nr linii', to słowo kluczowe GOTO może zostać pominięte (przyp. tłum.).

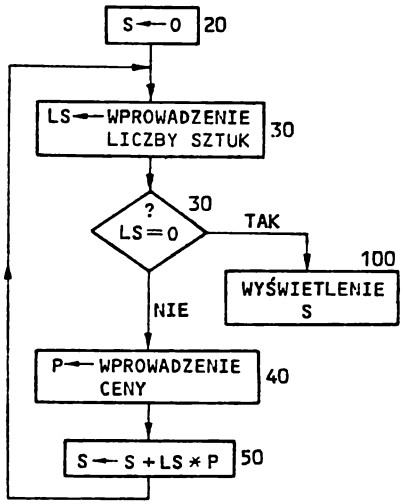
Wyjście z pętli

Instrukcja **IF... THEN...** *umożliwia wyjście z pętli.*

Poniższy program oblicza sumę wydatków.

```
10 S=0
20 INPUT "Sztuk? ";LS
30 IF LS=0 THEN GOTO 100
40 INPUT "Cena?  ";P
50 S=S+LS*P
60 GOTO 20
100 PRINT "Ogolna suma: ";S
110 END

RUN

Sztuk? 3
Cena?  50
Sztuk? 1
Cena?  12
Sztuk? 0

Ogolna suma: 162
```

Aby zakończyć program



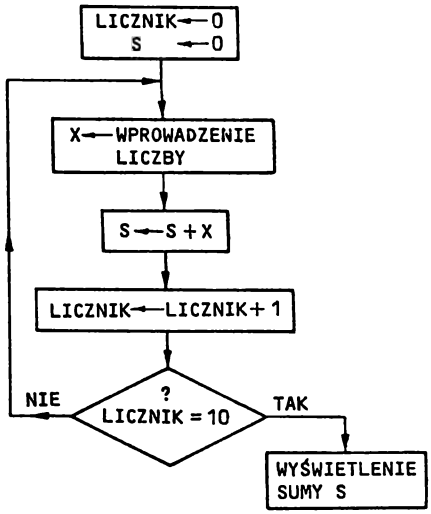
Po zsumowaniu wszystkich wydatków, w odpowiedzi na zapytanie 'Sztuk?' wprowadzamy wartość 0.

Wykonywane w 30-tej instrukcji badanie, czy wprowadzona wartość jest równa 0, umożliwia wyjście z pętli i przejście do wyświetlenia obliczonej sumy.

Wyjście z pętli z wykorzystaniem licznika

Inny sposób wyjścia z pętli polega na zliczaniu liczby przejść przez pętlę w specjalnej zmiennej (np. LICZNIK). Badanie wartości zmiennej LICZNIK umożliwia opuszczenie pętli.

```
10 LICZNIK=0
20 S=0
30 :
40 INPUT "Liczba? ";X
50 S=S+X
60 LICZNIK=LICZNIK+1
70 IF LICZNIK=10 THEN 100
80 GOTO 40
90 :
100 PRINT "Suma: ";S

RUN↵

Liczba? 12↵
Liczba? 7↵
Liczba? 13↵      } 10 razy
...
Liczba? 15↵

Suma: 142
```



Jeżeli nie jest wstępnie znana liczba wartości, które mają być zsumowane, wystarczy napisać:

```
 5 INPUT "Ile skladnikow? ";LS

70 IF LICZNIK=LS THEN 100

RUN↵

Ile skladnikow? 4↵

Liczba? 12↵
Liczba? 6↵       } 4 razy
Liczba? 8↵
Liczba? 13↵

Suma: 39
```

Badanie warunków złożonych

Warunki złożone mogą być badane za pomocą **operatorów logicznych AND** *i* **OR.**

warunek 1 AND warunek 2: Instrukcje występujące po THEN są wykonywane tylko wówczas, gdy jest spełniony warunek 1 I warunek 2.

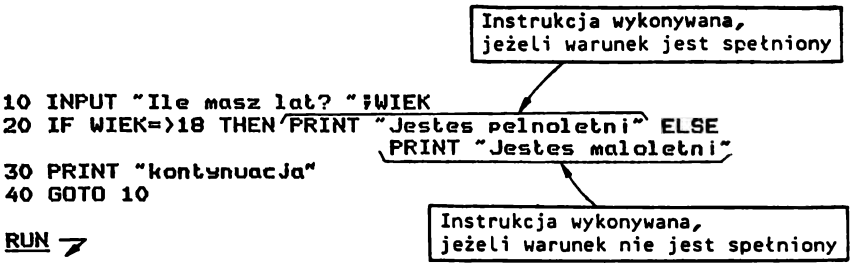
```
10 INPUT "Podaj liczbe? ";N
20 IF (N>0) AND (N<10) THEN PRINT "Ta liczba jest
                          wieksza od 0 i mniejsza od 10"
30 GOTO 10

RUN ↵

Podaj liczbe? 5↵
Ta liczba jest wieksza od 0 i mniejsza od 10
Podaj liczbe? 13↵
Podaj liczbe?
```

warunek 1 OR warunek 2: Instrukcje występujące po THEN są wykonywane wówczas, gdy jest spełniony warunek 1 LUB warunek 2 (lub oba warunki jednocześnie).

```
10 INPUT "Podaj dwie liczby X,Y ? ";X,Y
20 IF (X>0) OR (Y>0) THEN PRINT "X lub Y jest
                              dodatnia (lub obie)"
30 GOTO 10

RUN↵

Podaj dwie liczby X,Y ? 12,4↵
X lub Y jest dodatnia (lub obie)

Podaj dwie liczby X,Y ? -4,-1↵
```

IF warunek THEN instrukcje ELSE instrukcje (Microsoft 5.)

W wersji Basic Microsoft 5. występuje instrukcja **IF... THEN... ELSE...**, *umożliwiająca wykonanie określonych instrukcji w przypadku, gdy badany warunek nie jest spełniony.*



RUN ↵

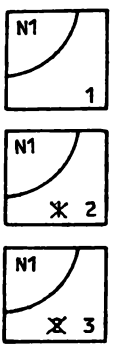
```
Ile masz lat? 25↵
Jestes pelnoletni
kontynuacja

Ile masz lat? 17↵
Jestes maloletni
kontynuacja
```

Pętla automatyczna FOR...NEXT

Przypuśćmy, że chcemy wyświetlić liczby 1, 2, 3 oraz ich kwadraty 1*1, 2*2, 3*3.

Pierwsze rozwiązanie (bez GOTO i IF... THEN...)

```
10 N1=1
20 PRINT N1,N1*N1
30 N1=N1+1
40 PRINT N1,N1*N1
50 N1=N1+1
60 PRINT N1,N1*N1

RUN

1         1
2         4
3         9
```



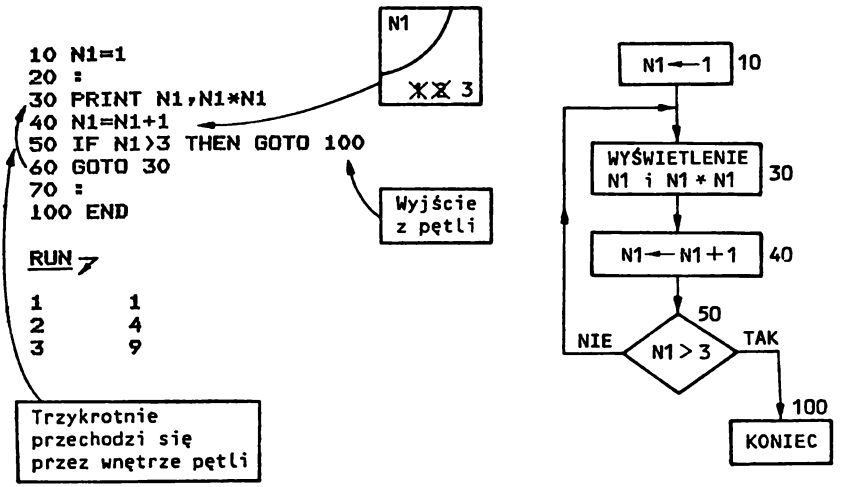
Stosowanie tej metody do wyznaczenia kwadratów liczb od 1 do 100 byłoby bardzo uciążliwe.

Drugie rozwiązanie (z GOTO i IF... THEN...)

Zamiast pisać wiele razy sekwencję:

N1 = N1 + 1
PRINT N1, N1*N1
napiszmy ją tylko raz.

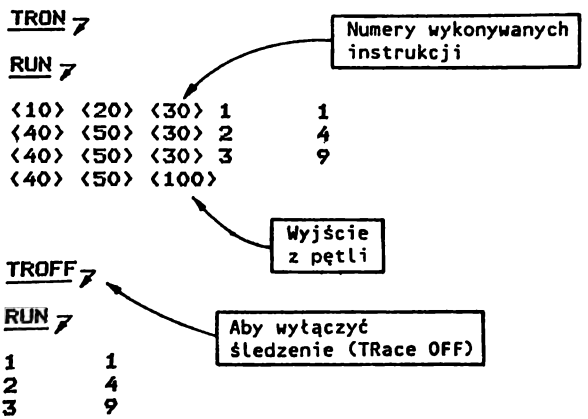
Aby ją wielokrotnie wykonać, użyjemy instrukcji skoku GOTO; badanie wartości N1 umożliwi nam wyjście z pętli. Kiedy N1 osiągnie wartość większą niż 3, zatrzymamy program.



Aby uzyskać listę kwadratów liczb od 1 do 100, wystarczy zmienić instrukcję 50:

```
50 IF N1=100 THEN GOTO 100
```

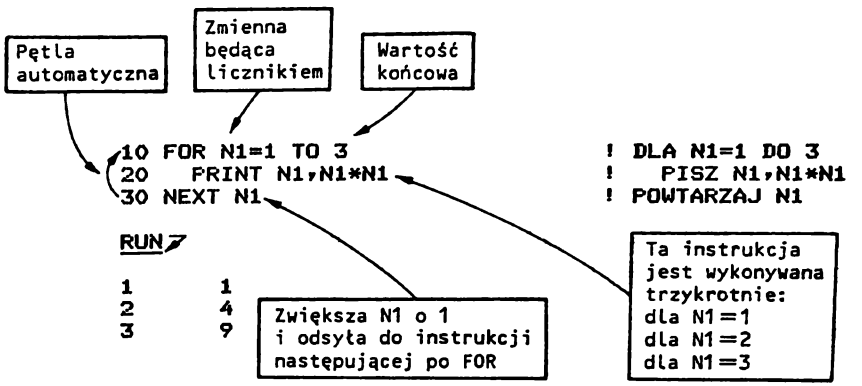
W celu prześledzenia przebiegu programu, użyjmy *zlecenia 'TRON'* (TRace ON). Wyświetlane będą numery wykonywanych instrukcji.



Uwaga: W komputerze APPLE II zamiast zlecenia TRON należy używać zlecenia TRACE, a zamiast TROFF – NOTRACE.

Trzecie rozwiązanie (pętla automatyczna FOR... NEXT)

Uprościmy poprzedni program stosując pętlę "automatyczną", czyli pętlę FOR... NEXT.



Jaki jest przebieg tego programu?

- Instrukcja 10 umieszcza w pamięci 'wartość końcową' 3 i nadaje 'licznikowi' N1 wartość 1.
- Instrukcja 20 jest wykonywana z wartością N1 = 1.
- Instrukcja '30 NEXT N1' zwiększa N1 o 1 i sprawdza, czy N1 jest mniejsze lub równe 'wartości końcowej' 3.
  Jeżeli N1 jest mniejsze lub równe 3, to jest wykonywana ponownie instrukcja następująca po FOR (20), tym razem z wartością N1 = 2.

W ten sposób instrukcja 20 jest wykonywana trzykrotnie, dla N1 = 1, 2, 3.

Przykłady

Wyświetlenie pięciu wierszy gwiazdek.

```
10 FOR N=1 TO 5
20   PRINT "**********"
30 NEXT N

RUN

**********
**********
**********
**********
**********
```

Wyświetlenie kwadratów liczb z przedziału 4—6.

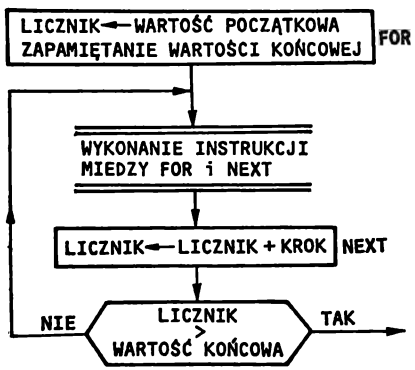
```
10 FOR N1=4 TO 6
20  PRINT N1,N1*N1
30 NEXT N1
40 PRINT "Koniec petli"

RUN↵

4       16
5       25
6       36
Koniec petli

TRON↵

RUN↵

<10> <20> 4       16
<30> <20> 5       25
<30> <20> 6       36
<30> <40> Koniec petli

TROFF↵
```

Pełna składnia pętli FOR

STEP umożliwia określenie kroku.

Specyfikacja 'STEP krok' oznacza, że za każdym razem, po wykonaniu instrukcji z wnętrza pętli, należy dodać do licznika wartość 'krok'.

Jeżeli opuści się specyfikację STEP, to po każdym przejściu przez wnętrze pętli licznik będzie zwiększany o 1. Tak więc pominięcie specyfikacji 'kroku' jest równoznaczne z nadaniem mu wartości 1.



FOR licznik = wartość początkowa TO wartość końcowa STEP krok
instrukcja 1
instrukcja 2
.....
NEXT licznik

Wykonując instrukcję FOR, Basic nadaje zmiennej będącej 'licznikiem' określoną 'wartość początkową' i zapamiętuje wskazaną 'wartość końcową' oraz 'krok'. Następnie są wykonywane wszystkie instrukcje występujące między instrukcjami FOR a NEXT, z wartością 'licznika' równą 'wartości początkowej'.

Wykonanie instrukcji NEXT zwiększa wartość 'licznika' o 'krok'[1].

- Jeżeli wartość 'licznika' jest nie większa niż 'wartość końcowa', to instrukcje między FOR a NEXT są wykonywane ponownie z nową wartością 'licznika'.
- Jeżeli wartość 'licznika' przekroczyła 'wartość końcową', to pętla kończy się, a wykonywanie programu jest kontynuowane od instrukcji następującej po instrukcji NEXT.

Przykłady

STEP wartość dodatnia

```
10 FOR C=1 TO 5 STEP 2
20  PRINT C;
30 NEXT C
40 PRINT "Koniec"
```

```
RUN ↵
```

```
1  3  5  Koniec
```

Tablica sinusów

```
 10 FOR KT=0 TO 3.14 STEP 3.14/10
 20  PRINT KT,SIN(KT)
 30 NEXT KT
```

```
RUN ↵
```

```
0               0
.314            .308866
.628            .587528
.942            .808736
```

[1] Zakładamy, że wartość kroku jest dodatnia (przyp. tłum.).

STEP wartość ujemna

```
10 FOR C=3 TO 1 STEP -1
20  PRINT C;
30 NEXT C

RUN ↵

3 2 1
```

Pętla FOR jest wykonywana co najmniej raz, nawet wówczas, gdy 'wartość końcowa' jest mniejsza niż 'wartość początkowa' (z wyjątkiem wersji Microsoft 5.).

```
10 D=3:F=1
20 :
30 FOR C=D TO F
40  PRINT C
50 NEXT C

RUN ↵

3
```

Pętla nie powinna być wykonana

Należy więc przeprowadzać dodatkowe badanie:

```
35 IF C>F THEN 50
```

Błędy

```
10 FOR I=1 TO 5
20  PRINT I
30 NEXT J

RUN ↵

1

NEXT WITHOUT FOR IN 30
```

Błąd! J zamiast I

Przykład

Tabliczka mnożenia przez 8:

Poniższa pętla FOR wyświetla tabliczkę mnożenia przez 8.

```
30 FOR N1=1 TO 10
40     PRINT N1;"X ";8;"= ";N1*8
50 NEXT N1

RUN

1 X 8 = 8
2 X 8 = 16
3 X 8 = 24
..........
10 X 8 = 80
```

Nie stosując pętli, musielibyśmy napisać:

```
 10 PRINT 1;"X ";8;"= ";1*8
 20 PRINT 2;"X ";8;"= ";2*8
 30 PRINT 3;"X ";8;"= ";3*8
 40 ......................
 90 PRINT 9;"X ";8;"= ";9*8
100 PRINT 10;"X ";8;"= ";10*8
```

Przyznasz chyba, że byłoby to zbyt uciążliwe!

Uogólnijmy powyższy program na wszystkie tabliczki mnożenia:

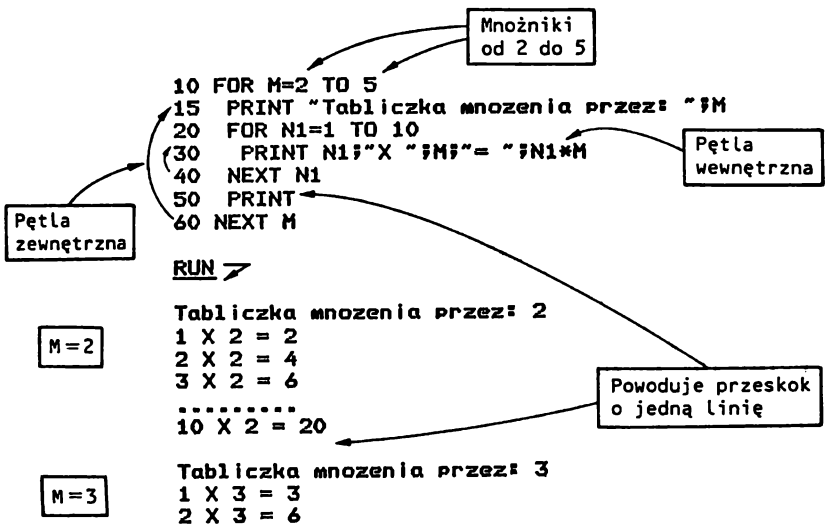
```
10 INPUT "Jaki mnoznik? ";M
20 :
30 FOR N1=1 TO 10
40  PRINT N1;"X ";M;"= ";N1*M
50 NEXT N1
60 GOTO 10

RUN

Jaki mnoznik? 9

1 X 9 = 9
2 X 9 = 18
```

Pętle zagnieżdżone

Pętle FOR mogą być zagnieżdżane.

Poniższy program wyprowadza tabliczki mnożenia dla mnożników z przedziału 2–5.

Wewnętrzna pętla jest wykonywana najpierw dla M = 2, następnie dla M = 3 itd.



Wykonaj ten program ze zleceniem TRON, co umożliwi ci lepsze zrozumienie jego przebiegu.

Przykład

Oto zegar wyświetlający godziny, minuty i sekundy:
Opóźnienie w linii 50 musi być dopasowane do twojego komputera.

```
10 FOR H=0 TO 23
20  FOR M=0 TO 59
30   FOR S=0 TO 59
40     PRINT H;"h. ";M;"m. ";S;"s."
50     FOR ZT=1 TO 170:NEXT ZT          :REM opoznienie
60   NEXT S
70  NEXT M
80 NEXT H
```

RUN↵

```
0 h. 0 m. 0 s.
0 h. 0 m. 1 s.
0 h. 0 m. 2 s.
```

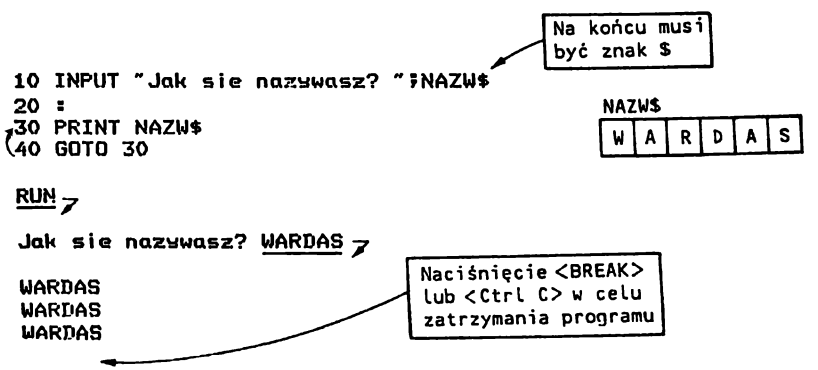
W komputerze TRS-80 możesz zastąpić 40-tą instrukcję przez:

```
PRINT @64*5+10,H;"h.";M;"m.";S;"s."
```

Teksty

W języku BASIC występują **zmienne typu tekstowego.** *Są one odróżniane od zmiennych numerycznych przez umieszczany na końcu nazwy zmiennej znak* **$.**

Zmienne te są stosowane w programach z różnorodnych dziedzin (zarządzanie, nauczanie, gry).



Aby zablokować przeskok do nowej linii po wyprowadzeniu nazwiska, wystarczy dopisać **średnik** *za* **NAZW$.**

```
30 PRINT NAZW$; ↵
RUN ↵
WARDASWARDASWARDAS
```

Przydzielenie zmiennej NAZW$ wartości 'WARDAS' wymaga ujęcia tekstu w znaki cudzysłowu. W przeciwnym przypadku 'WARDAS' byłoby interpretowane jako nazwa zmiennej.

```
10 NAZW$="WARDAS"
20 PRINT NAZW$
```

Znaki cudzysłowu są niezbędne

```
RUN↵

WARDAS
```

W BASIC Microsoft długość tekstu jest ograniczona do 255 znaków.

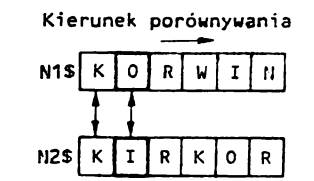
Porównywanie tekstów

Basic porównuje dwa teksty znak po znaku, od lewej do prawej strony[1].

Porównywanie jest przerywane, kiedy znak jednego tekstu okaże się różny od odpowiadającego mu znaku w drugim tekście[2].

```
10 INPUT "Pierwsze nazwisko ? ";N1$
20 INPUT "Drugie nazwisko?    ";N2$
30 :
40 IF N1$>N2$ THEN PRINT N1$;" jest wieksze od ";N2$
50 GOTO 10
```

```
RUN↵

Pierwsze nazwisko? KORWIN↵
Drugie nazwisko?   KIRKOR↵

KORWIN jest wieksze od KIRKOR
```



Kiedy jeden znak okaże się większy od drugiego, porównywanie jest przerywane (O jest większe od I)

Uwaga! Basic przeznacza dla tekstów specjalny obszar pamięci. Obszar ten w większości systemów obejmujący 1000 znaków, w komputerze TRS-80 jest ograniczony do 50 znaków.

---

[1] Tzn. porównuje kody ASCII kolejnych znaków obu tekstów — patrz dodatek IV (przyp. tłum.).
[2] Albo gdy dojdzie się do końca jednego z tekstów. W takim przypadku krótszy tekst jest uważany za "mniejszy" (przyp. tłum.).

Należy to uwzględnić na początku programu uruchamianego w systemie TRS-80.

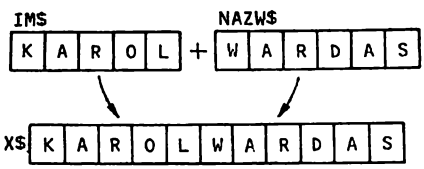
```
10 CLEAR 2000
```

Rezerwuje 2000 znaków na teksty

Konkatenacja tekstów

Konkatenacja *(składanie) tekstów jest wykonywana za pomocą operatora* '+' (tak, jak przy dodawaniu zmiennych numerycznych).

```
10 INPUT "Imie?      ";IM$
20 INPUT "Nazwisko? ";NAZW$
30 :
40 X$=IM$+NAZW$
50 Y$=IM$+" "+NAZW$
60 PRINT X$
70 PRINT Y$

RUN↵

Imie?      KAROL↵
Nazwisko? WARDAS↵

KAROLWARDAS
KAROL WARDAS
```



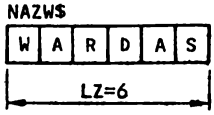
X$ = IM$ + NAZW$

Y$ = IM$+" "+NAZW$

Długość tekstu: LEN (tekst)

LEN (tekst) *podaje liczbę znaków tekstu.*

```
10 INPUT "Twoje nazwisko? ";NAZW$
20 LZ=LEN(NAZW$)
30 PRINT "Twoje nazwisko zawiera "; LZ;"znakow"

RUN↵

Twoje nazwisko? WARDAS↵

Twoje nazwisko zawiera 6 znakow
```

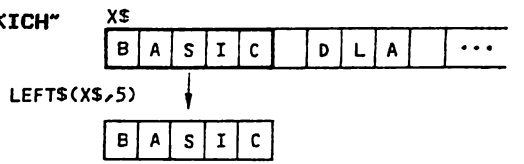


Długość zmiennej tekstowej może się zmieniać podczas wykonywania programu. Na początku programu nie trzeba rezerwować obszaru dla maksymalnej długości zmiennej.

LEFT$ (tekst, liczba znaków do pobrania)

Wydziela grupę znaków z lewej strony tekstu.

```
10 X$="BASIC DLA WSZYSTKICH"
20 Y$=LEFT$(X$,5)
30 PRINT Y$

RUN↵

BASIC
```



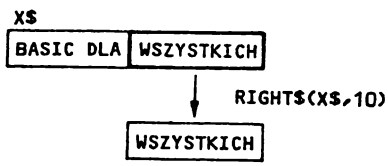
Na przykład:

```
10 INPUT "Twoje nazwisko? ";NAZW$
20 FOR L=1 TO LEN(NAZW$)
30     PRINT LEFT$(NAZW$,L)
40 NEXT L

RUN↵

Twoje nazwisko? WARDAS↵

W
WA
WAR
WARD
WARDA
WARDAS
```

RIGHT$ (tekst, liczba znaków do pobrania)

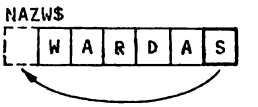
Wydziela grupę znaków z prawej strony tekstu.

```
10 X$="BASIC DLA WSZYSTKICH"
20 PRINT RIGHT$(X$,10)

RUN↵

WSZYSTKICH
```



Poniższy program wyprowadza cykliczne permutacje tekstu:

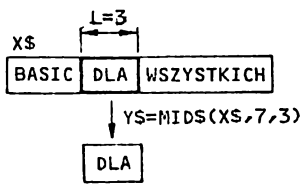
```
10 NAZW$="WARDAS"
20 PRINT NAZW$
30 FOR N=1 TO LEN(NAZW$)
40   NAZW$=RIGHT$(NAZW$,1)+LEFT$(NAZW$,LEN(NAZW$)-1)
50   PRINT NAZW$
60 NEXT N

RUN

WARDAS
SWARDA
ASWARD
DASWAR
RDASWA
ARDASW
WARDAS
```



MID$ (tekst, pozycja początkowa, liczba znaków do pobrania)

Wydziela grupę znaków z wnętrza tekstu.

```
10 X$="BASIC DLA WSZYSTKICH"
20 Y$=MID$(X$,7,3)
30 PRINT Y$

RUN

DLA
```



Na przykład:

```
10 INPUT "Nazwisko? ";NAZW$
20 FOR P=1 TO LEN(NAZW$)
30   PRINT MID$(NAZW$,P,1)
40 NEXT P

RUN

Nazwisko? WARDAS

W
A
R
D
A
S
```

Przeskok do nowej linii, ponieważ nie ma znaku;

Poniższy program wyświetla nazwisko wspak:

```
10 INPUT "Nazwisko? ";NAZW$
20 FOR P=LEN(NAZW$) TO 1 STEP -1
30   PRINT MID$(NAZW$,P,1);
40 NEXT P

RUN↵

Nazwisko? KOWALCZYK↵

KYZCLAWOK
```

Ten natomiast wydziela poszczególne słowa zdania:

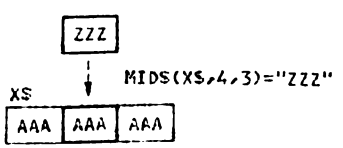
```
10 INPUT "Wprowadz zdanie: ";ZD$
20 Y$=""
30 :
40 FOR P=1 TO LEN(ZD$)
50   X$=MID$(ZD$,P,1)
60   IF X$<>" " THEN Y$=Y$+X$
70   IF X$=" " THEN PRINT Y$:Y$=""
80 NEXT P
90 PRINT Y$

RUN↵

Wprowadz zdanie: KOTECZEK CICHO MRUCZY↵

KOTECZEK
CICHO
MRUCZY
```

MID$ (tekst, pozycja początkowa, liczba) = tekst zastępczy

(Microsoft 5., TRS-80)

Ta postać **MID$** *umożliwia wymienienie grupy znaków wewnątrz tekstu.*

```
10 X$="AAAAAAAAA"
20 MID$(X$,4,3)="ZZZ"
30 PRINT X$

RUN↵

AAAZZZAAA
```



Długość tekstu zastępczego nie może być większa od długości wymienianego ciągu znaków.

Program wymiany znaków dla APPLE II

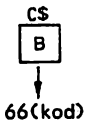
```
 10 X$="AAAAAAAAA"
 20 P=4:Y$="ZZZ"
 30 GOSUB 100
 40 PRINT X$
 50 END
 60 :
100 X$=LEFT$(X$,P-1)+Y$+RIGHT$(X$,LEN(X$)-P-LEN(Y$)+1)
110 RETURN
```

ASC (znak)

Podaje kod znaku.

Wewnątrz komputera wszystkie znaki są reprezentowane w formie binarnej. Programista może uzyskać kody znaków w postaci liczb dziesiętnych.

```
10 C$="B"
20 PRINT ASC(C$)

RUN ↵
66
```



Kody kolejnych liter alfabetu (A, B, C, ... Z): 65, 66,67 ...90

ASC (tekst)

Podaje kod pierwszego znaku tekstu.

Tekst nie może być pusty.

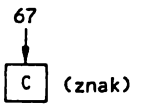
```
10 X$="CHARLIE"
20 PRINT ASC(X$)

RUN ↵
67
```

CHR$ (kod)

Dostarcza znak o podanym kodzie.

Nie wszystkie znaki są znakami graficznvmi.

```
10 X=67
20 PRINT CHR$(X)

RUN↵
```




```
C

10 FOR K=65 TO 65+25
20  PRINT CHR$(K);
30 NEXT K

RUN↵

ABCD.....Z
```

Poniższy program zastępuje każdą literę kolejną literą alfabetu (na przykład A jest zastępowane przez B).

```
10 INPUT "Nazwisko? ";NAZW$
20 :
30 FOR P=1 TO LEN(NAZW$)
40  X=ASC(MID$(NAZW$,P,1))
50  IF X=90 THEN X=64        :REM--- Dla litery Z ---
60  PRINT CHR$(X+1);
70 NEXT P

RUN↵

Nazwisko? WARDAS↵
ZBSEBT
```

Zakodowane nazwisko

Pewne znaki służą do przesyłania poleceń do urządzeń zewnętrznych:

Kilka kodów ASCII

PRINT CHR$ (10); powoduje przejście do nowej linii
(bez powrotu do początku linii)
PRINT CHR$ (13); powoduje powrót do początku linii
(bez przejścia do nowej linii)

Dla TRS-80

PRINT CHR$ (23); zwiększa dwukrotnie rozmiar znaku.

Jeżeli twój komputer jest wyposażony w generator dźwięku, to jest on uruchamiany przez:

```
PRINT CHR$(7);
```

Aby uzyskać dłuższy sygnał, należy napisać:

```
10 FOR T=1 TO 100
20 PRINT CHR$(7);
30 NEXT T
```

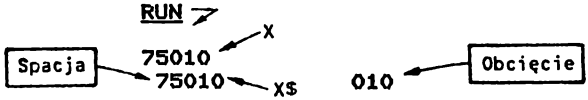
Znak średnika umieszczony za CHR$ (7) zapobiega zbędnym przeskokom kursora do kolejnych linii ekranu.

STR$ (X)

Przekształca liczbę do postaci tekstowej.

Umożliwia to stosowanie funkcji LEFT$, RIGHT$, MID$ w celu wydzielenia poszczególnych cyfr.

```
10 X=75010
20 X$=STR$(X)
30 PRINT X
40 PRINT X$,RIGHT$(X$,3)

RUN
 75010
 75010
```



Zauważ, że pierwszym znakiem wyprowadzonego tekstu jest spacja, odpowiadająca niewyświetlanemu symbolowi +.

VAL (tekst)

Wyznacza wartość numeryczną tekstu rozpoczynającego się od ciągu cyfr (lub od spacji).

Jeżeli tekst rozpoczyna się od litery, to jego wartość jest równa zeru.

```
10 CENA$="22 DOLLARS"
20 X=VAL(CENA$)
30 PRINT X

RUN↵

22
```

Tekst nie może być traktowany bepośrednio jako wartość.

```
10 CENA$="22"
20 PRINT CENA$*3
```

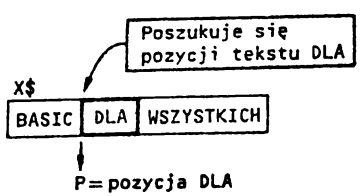
Instrukcja 20 jest błędna.

INSTR (początkowa pozycja poszukiwań, tekst, tekst szukany)

(Microsoft 5.)

Wskazuje pozycję tekstu we wnętrzu innego tekstu.

```
10 X$="BASIC DLA WSZYSTKICH"
20 P=INSTR(X$,"DLA")
30 PRINT P

RUN ↵

7
```



Gdy 'początkowa pozycja poszukiwań' nie zostanie określona, wówczas przyjmuje ona standardową wartość 1.
Jeżeli tekst nie zostanie odnaleziony, to P będzie równe 0.

Zbadamy, czy podane przez operatora imię należy do określonego zbioru:

```
10 ZI$="JAN PIOTR PAWEL JACEK"
20 :
30 INPUT "Imie? ";IM$
40 P=INSTR(ZI$,IM$)
50 IF P=0 THEN PRINT "Sprobuj jeszcze raz":GOTO 30
60 REM -- dalszy ciag programu
```

Jeżeli operator wprowadzi imię różne niż JAN, PIOTR, ..., to program je odrzuci.

Podprogram zastępujący INSTR

```
 10 CLEAR 1000
 20 X$="BASIC DLA WSZYSTKICH"
 30 :
 40 INPUT "Poszukiwana sekwencja znakow? ";SZ$
 50 GOSUB 100
 60 IF P=0 THEN PRINT "Sekwencji nie odnaleziono":GOTO 40
 70 PRINT "Sekwencja odnaleziona na pozycji: ";P:GOTO 40
 90 REM ------------------------------------------------
100 L=LEN(SZ$)
110 FOR P=1 TO LEN(X$)-L
120  IF MID$(X$,P,L)=SZ$ THEN RETURN
130 NEXT P
140 P=0:RETURN
```

```
SZ$
┌─────┐
│ DLA │  Poszukiwany tekst
└─────┘

X$
┌──────────────────┐
│ BASIC DLA WSZYSTKICH │
└──────────────────┘

 ├─┤
  ├─┤        Porównywanie
   ├─┤       sukcesywne
    ├─┤
```

Uwaga: Tekst wprowadzany w odpowiedzi na INPUT nie może zawierać przecinka, gdyż pełni on funkcję separatora.

```
10 INPUT "Imiona? ";IM$
```

RUN ↵

```
Imiona? ANNA MARIA
```

Gdyby operator wprowadził przecinek po słowie Anna, wówczas wszystkie znaki występujące po tym słowie zostałyby pominięte.

Uruchamianie programów: BREAK-CONT-STOP

Programy nie zawsze działają poprawnie od pierwszego razu. Basic wysyła wprawdzie komunikaty o niektórych błędach (na przykład o błędach składniowych), ale nie wykrywa błędów logicznych.
W bardziej subtelnych przypadkach należy prześledzić kolejne etapy przebiegu programu, co w systemie Basic jest zadaniem dość prostym.

BREAK-CONT

Naciskając klawisz <BREAK> (lub <Ctrl C>) wstrzymujemy wykonywanie programu.

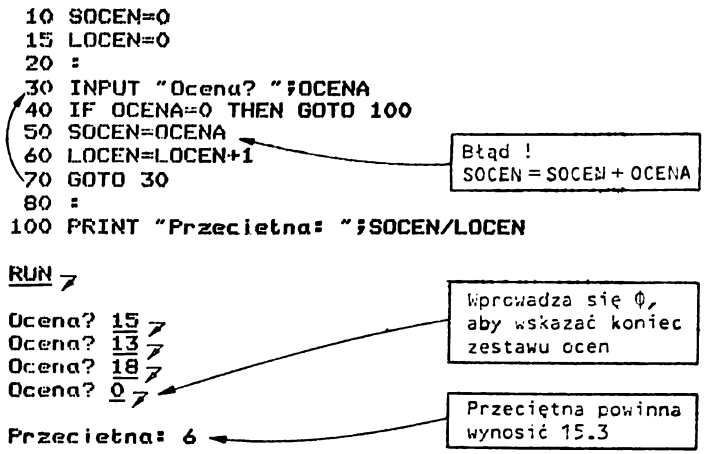
Możemy wówczas wyświetlić na ekranie, w trybie bezpośrednim, wartości poszczególnych zmiennych.

Wykonywanie przerwanego programu można wznowić pisząc CONT (od ang. CONtinue – wznów).

Przykład

Program na str. 61 wyznacza średnią z wielu ocen w skali punktowej 0–20.

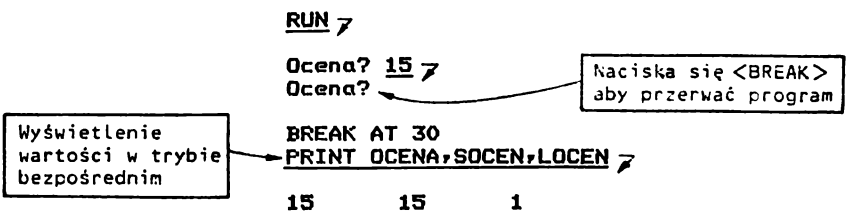
Popełniliśmy (celowo!) błąd: W linii 50, zamiast SOCEN = SOCEN + OCENA, napisaliśmy SOCEN = OCENA.



Wyznaczona średnia jest oczywiście nieprawidłowa.

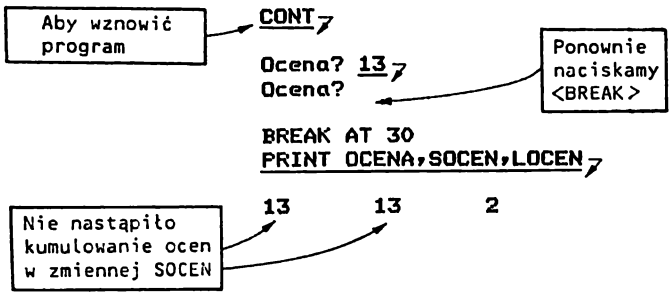
Uwaga: Instrukcje 10 i 15, zerujące wstępnie zmienne SOCEN i LOCEN, nie są niezbędne, gdyż RUN zeruje wszystkie zmienne programu. Zalecane jest jednak wykonywanie tego zerowania w przypadku bardziej złożonego programu, w którym zmienne SOCEN i LOCEN mogą mieć wartości niezerowe, ze względu na możliwość uprzedniego używania tych zmiennych w innej części programu.

Wykonajmy program powtórnie i przerwijmy go po wprowadzeniu pierwszej oceny. Możemy teraz wyprowadzić w trybie bezpośrednim wartości zmiennych OCENA, SOCEN i LOCEN.



Na razie wszystko jest w porządku.

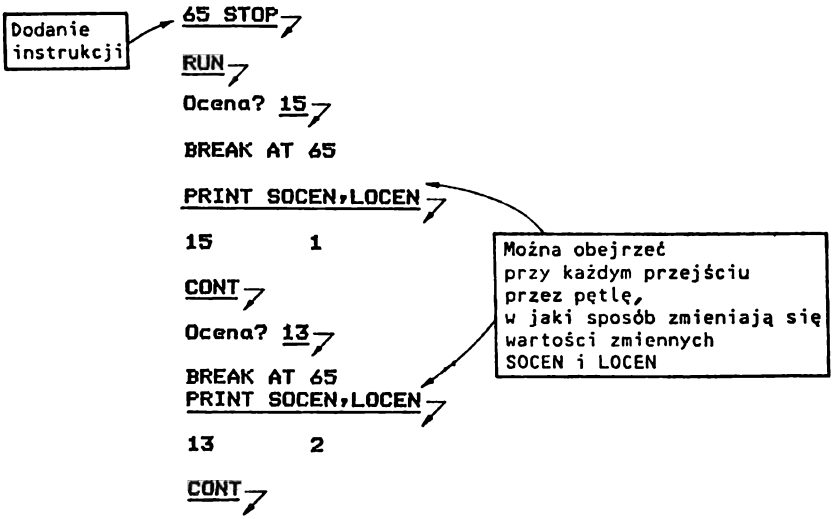
Napiszmy CONT w celu wznowienia wykonywania programu i przerwijmy program ponownie po wprowadzeniu drugiej oceny:



Analizując wartość zmiennej SOCEN stwierdzamy, że nie nastąpiło sumowanie ocen.

STOP

Zamiast naciskać klawisz <BREAK>, możemy dołączyć do programu instrukcję STOP z numerem 65.



W komputerze APPLE II, program przerwany przez <Ctrl C> na instrukcji INPUT nie może być wznowiony. Należy więc wstawić do programu instrukcję STOP.

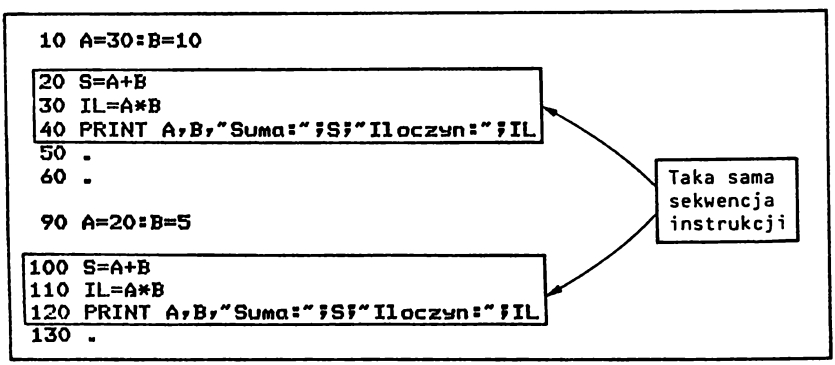
Gdy program się **"zapętli"**

Jeżeli program się nie kończy – naciśnij <BREAK>, a następnie napisz TRON i CONT. W ten sposób można zlokalizować tę część programu, w której występuje pętla.

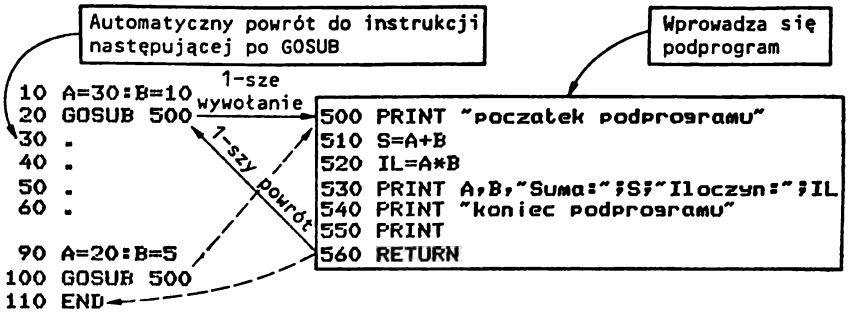
Podprogramy GOSUB/RETURN

Często zdarza się, że ta sama sekwencja instrukcji musi być użyta w programie wiele razy.

Podprogram umożliwia **jednokrotne** *napisanie tej sekwencji, którą wystarczy wywołać z różnych miejsc programu instrukcją* **GOSUB** *nr instrukcji.*



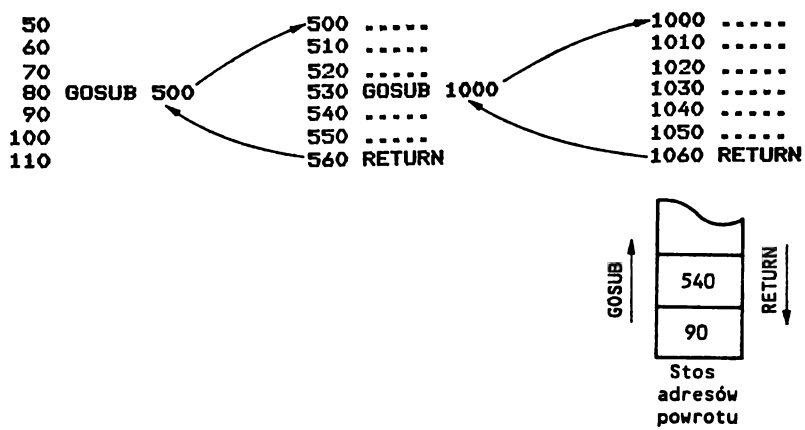
'20 GOSUB 500' powoduje przejście do linii o numerze 500 (tak jakby to zrobiła instrukcja 'GOTO 500'), *z tym że umieszczona na końcu podprogramu instrukcja* **RETURN** *(WRÓĆ) powoduje* **automatyczny powrót** *do instrukcji następującej po* **'GOSUB 500'**, czyli w tym przykładzie do instrukcji 30.
Przy drugim wywołaniu podprogramu z linii 100, następuje powrót do linii 110.




```
RUN

początek podprogramu
30        10        Suma:40  Iloczyn:300


początek podprogramu
20        5         Suma:25  Iloczyn:100
koniec podprogramu
```

Podprogram może wywoływać inny podprogram.

Basic zarządza adresami powrotnymi (w tym przykładzie 90 i 540) za pomocą stosu.



Instrukcje są wykonywane w następującej kolejności:

```
50 60 70 80
500 510 520 530                    Początek pierwszego podprogramu
1000 1010 ... 1050 1060            Drugi podprogram
540 550 560                        Koniec pierwszego podprogramu
90 100                             Powrót do programu głównego
```

Nie próbujcie wracać z podprogramu ani też do niego wchodzić za pomocą GOTO!

Interludium

W sprawie schematów blokowych

Czy należy stosować schematy blokowe?

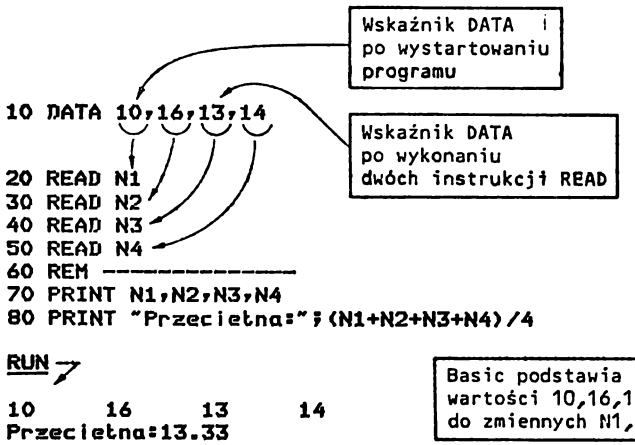
Sprawa ta jest bardzo kontrowersyjna; niektórzy są ich zwolennikami, inni natomiast nie stosują ich wcale.

Zwolennicy utrzymują, że właściwe programowanie powinno być poprzedzone opracowaniem schematu blokowego, że sam program nigdy nie jest dostatecznie przejrzysty.

Ci natomiast, którzy ich nie stosują, twierdzą, że w przypadku dobrej strukturalizacji programu przygotowywanie schematu blokowego jest zbędne, że nadmiernie rozbudowany schemat szybko staje się zbyt pogmatwany.
Naszym zdaniem, *należy dokładać wielu starań w celu opracowywania programów* w taki sposób, aby ich zrozumienie wymagało minimalnego wysiłku.
Schemat blokowy rzadko kiedy "podąża" za programem; pozostaje tylko sam program. *Z tego względu program powinien być udokumentowany jak tylko można najlepiej.*

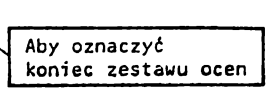
DATA/READ/RESTORE

Instrukcja **DATA** *umożliwia definiowanie danych w samym programie. Dane te są następnie wczytywane do zmiennych za pomocą instrukcji* **'READ zmienna'** *(CZYTAJ zmienna).*



'READ N1' wczytuje pierwszą daną (10) do zmiennej N1. Wskaźnik DATA (którym zarządza system Basic) jest zwiększany o 1.
Tak więc 'READ N2' wczytuje drugą daną do N2, itd...
W powyższym przykładzie przyjęliśmy założenie, że liczba odczytywanych danych (4) jest znana.
Aby oznaczyć koniec danych, umieścimy na końcu listy liczbę 99.

```
 10 DATA 10,16,13,15,18,99
 30 READ OCENA
 40 IF OCENA=99 THEN GOTO 100
 45 PRINT OCENA,
 50 SOCEN=SOCEN+OCENA
 60 LOCEN=LOCEN+1
 70 GOTO 30
 80 REM --------------------
100 PRINT "Przecietna:";SOCEN/LOCEN

RUN

10     16     13     15     18     Przecietna:14.6
```



Dane mogą być zapisywane w wielu liniach:

```
10 DATA 10,16,13
20 DATA 15,18,99
```

jest równoważne linii 10 przedstawionego wyżej programu.

Znaki specjalne

Dane DATA zawierające znaki specjalne muszą być ujmowane w znaki **cudzysłowu**[1].
Bez znaków cudzysłowu, przecinek byłby potraktowany jako separator.

```
10 DATA "Anna, Maria"
20 REM --------------
30 READ X$
40 PRINT X$

RUN

Anna, Maria
```

[1] W znaki cudzysłowu muszą być ujmowane dane typu tekstowego zawierające przecinki, dwukropki lub znaczące spacje na początku lub końcu (przyp. tłum.).

Liczba odczytywana jako tekst

Jeżeli wartość numeryczna ma być odczytywana jako ciąg znaków, to musi być ona ujęta w znaki **cudzysłowu:**

```
10 DATA "75000"
20 READ X$
30 PRINT X$,LEFT$(X$,2)

RUN
 75000       75
```

Błędy

OUT OF DATA: Jeżeli liczba wykonanych instrukcji READ przekracza liczbę danych zdefiniowanych w DATA, to wyświetlany jest komunikat OUT OF DATA.

```
10 DATA 15,10
20 READ X
30 READ Y
40 READ Z
```

Brakuje danej

TYPE MISMATCH: Typ odczytywanej danej musi być zgodny z typem zmiennej.

```
10 DATA PIEKARSKI
20 READ X

RUN

TYPE MISMATCH
```

Tekst

Zmienna numeryczna

Dana (typ tekstowy) nie jest zgodna z typem zmiennej (typ numeryczny)

Należy napisać: 20 READ X$

RESTORE

Ustawia wskaźnik na początku DATA, co umożliwia ponowne odczytanie zestawu danych.

```
 10 DATA 6,3,14
 20 :
 30 READ A
 40 READ B
 50 READ C
 60 :
 70 RESTORE
 80 :
 90 READ D,E,F
100 :
110 PRINT A,B,C
120 PRINT D,E,F

RUN↵

6       3       14
6       3       14
```

RESTORE nr linii

Ustawia wskaźnik na określonę linię DATA, co umożliwia wybranie konkretnego fragmentu zestawu danych.
Przedstawiamy poniżej przykłady kilku programów wykorzystujących instrukcję DATA.

Przykłady

Spis telefonów

Spis telefonów jest zdefiniowany za pomocą instrukcji DATA. Po podaniu nazwiska uzyskuje się odpowiedni numer telefonu.

```
10 REM SPT      SPIS TELEFONOW
15 :
20 CLEAR 1000
30 DATA "BRANICKI",    44-22-11
40 DATA "BARANOWSKI",  17-21-47
45 DATA "CICHOCKI",    19-50-85
50 DATA *
60 REM ----------------------------------------
70 PRINT
80 INPUT "Nazwisko? ";NAZW$
85 :
90 RESTORE      :REM Ustawienie na poczatek danych
100 :
110 READ N$:IF N$="*" THEN PRINT "Brak abonenta":GOTO 70
120 READ TEL$
130 IF N$=NAZW$ THEN PRINT:PRINT N$,TEL$:GOTO 70
140 GOTO 110
```

```
RUN ↵
Nazwisko? CICHOCKI ↵
CICHOCKI        19-50-85
```

Modyfikacja linii 130 umożliwia uzyskanie numeru po wprowadzeniu początkowego fragmentu nazwiska:

```
130 IF NAZW$=LEFT$(N$,LEN(NAZW$)) THEN PRINT N$,TEL$:GOTO 70
```

```
RUN↵
Nazwisko? BRA↵
BRANICKI        44-22-11
```

Jeżeli kilka nazwisk zaczyna się na 'BRA', to otrzymamy oczywiście pierwszy numer z listy.

Uporządkowany alfabetycznie spis telefonów można uzyskać wczytując dane DATA do dwóch tablic, które następnie należy posortować (patrz rozdz. "Tablice").

Budżet domowy

W każdej linii DATA są podawane kolejno miesięczne wydatki: czynsz, żywność, wydatki różne. Program odczytuje te dane i oblicza sumę wydatków w każdym miesięcu oraz sumę wydatków poszczególnych typów.

```
10 REM  BUDD              BUDZET DOMOWY
20 :                  Czynsz Żywność  Wydatki
25 :                                  różne
30 CLEAR 1000           ↓      ↓       ↓
40 DATA STYCZEN,  2000, 18000, 10000
50 DATA LUTY    ,  2000, 15000, 17000
60 DATA MARZEC  ,  2000, 16000,  8000
70 DATA *
80 REM ------------------------------
90 SCZYNSZ=0:SZYWN=0:SROZNE=0
100 :
110 LPRINT ,"CZYNSZ","ZYWNOSC","ROZNE"
120 :
130 FOR MIES=1 TO 12
140  READ MIES$:IF MIES$="*" THEN 230
150  READ CZYNSZ
160  READ ZYWN
170  READ ROZNE
180  SMIES=CZYNSZ+ZYWN+ROZNE
190  SCZYNSZ=SCZYNSZ+CZYNSZ:SZYWN=SZYWN+ZYWN:SROZNE=SROZNE+ROZNE
200  LPRINT MIES$,CZYNSZ,ZYWN,ROZNE,SMIES
210 NEXT MIES
220 :
230 LPRINT                                  LPRINT wyprowadza
240 LPRINT ,SCZYNSZ,SZYWN,SROZNE  <-------  wyniki na drukarkę
```

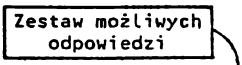
	CZYNSZ	ZYWNOSC	ROZNE	
STYCZEN	2000	18000	10000	30000
LUTY	2000	15000	17000	34000
MARZEC	2000	16000	8000	26000
	6000	49000	35000	

Pytania/Odpowiedzi

Program stawia pytanie, podaje kilka rozwiązań do wyboru, a następnie sprawdza poprawność odpowiedzi.

```
10 REM PTO     PYTANIA/ODPOWIEDZI (z podawaniem mozliwych rozwiazan)
20 :
30 :                                                        Poprawna
40 CLEAR 1000     Pytanie                Proponowane odpowiedzi   odpowiedź
50 DATA "Stolica HISZPANII?     ","MADRYD BRUKSELA MEDIOLAN","MADRYD"
60 DATA "Data HOLDU PRUSKIEGO? ","1515 1525 1410"          ,"1525"
70 DATA *
80 :
90 READ P$:IF P$="*" THEN STOP    :REM pobranie pytania
100 READ PO$                      :REM proponowane odpowiedzi
110 READ DO$                      :REM dobra odpowiedz
120 :
130 PRINT:PRINT P$;PO$            :REM Pytanie/proponowane odpowiedzi
140 :
150 INPUT "Odpowiedz? ";ODP$:IF ODP$="" THEN 150
160 :
170 IF ODP$=DO$ THEN PRINT:PRINT "Dobrze":GOTO 90
180 PRINT:PRINT "Prawidlowa odpowiedz: ";DO$
190 GOTO 90
RUN↵

Stolica HISZPANII?    MADRYD BRUKSELA MEDIOLAN
Odpowiedz? MADRYD↵

Dobrze
```

Pytania/Odpowiedzi (z użyciem INSTR)



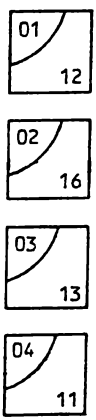

```
10 REM PO    PYTANIA/ODPOWIEDZI (z wykorzystaniem INSTR)
20 :
30 CLEAR 1000
40 DATA "Stolica HISZPANII?       ","MADRYD"
50 DATA "Panstwo BENELUXU?        ","BELGIA HOLANDIA LUKSEMBURG"
60 DATA "Rok BITWY POD WIEDNIEM? ","1683"
70 DATA *
80 :
100 READ P$:IF P$="*" THEN STOP
110 READ O$
120 :
130 PRINT:PRINT P$;
140 INPUT "Twoja odpowiedz? ";ODP$:IF ODP$="" THEN 140
150 :
160 P=INSTR(O$,ODP$)
170 IF P=0 THEN PRINT:PRINT "Odpowiedz: ";O$:GOTO 100
180 PRINT:PRINT "Dobrze":PRINT
190 GOTO 100
200 '
RUN
```

```
Stolica HISZPANII ?       Twoja odpowiedz? MADRYD

Dobrze
```

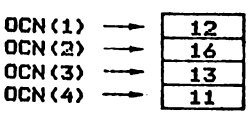
Tablice

Do zapamiętania czterech ocen moglibyśmy użyć czterech zmiennych N1, N2, N3, N4:

```
10 INPUT "Ocena1? ";01
20 INPUT "Ocena2? ";02
30 INPUT "Ocena3? ";03
40 INPUT "Ocena4? ";04
```



Użyjemy jednak **tablicy**, którą nazwiemy OCN( ).
Elementy tablicy OCN( ) są identyfikowane przez nazwy OCN(1), OCN(2), OCN(3), OCN(4).



Tablica OCN()

Moglibyśmy napisać jak niżej:

```
10 INPUT "Ocena1? ";OCN(1)
20 INPUT "Ocena2? ";OCN(2)
30 INPUT "Ocena3? ";OCN(3)
40 INPUT "Ocena4? ";OCN(4)
```

Lepiej jest jednak użyć pętli FOR; zmieniający się indeks umożliwia skrócenie tekstu programu.

```
      10 FOR N=1 TO 4
4 razy( 20    INPUT "Ocena? ";OCN(N)
      30 NEXT N
```

Za pierwszym razem równoważne OCN(1), ponieważ N=1

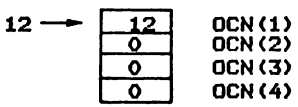
Początkowo N jest równie 1.
Tak więc

```
20 INPUT "Ocena? ";OCN(N)
```

jest równoważne

```
20 INPUT "Ocena? ";OCN(1)
```

Pierwsza ocena jest więc wprowadzana do OCN(1).



Przy drugim przejściu przez pętlę FOR, OCN(N) jest równoważne OCN(2), ponieważ N przyjęło wartość 2.

```
RUN↵

Ocena? 12↵
Ocena? 16↵
Ocena? 13↵
Ocena? 11↵

PRINT OCN(1)+OCN(2)+OCN(3)+OCN(4)↵
52
```

Suma ocen

Napisz w trybie bezpośrednim

```
PRINT OCN(1),OCN(2),OCN(3),OCN(4)↵

12          16          13          11
```

Nie stosując pętli FOR, moglibyśmy napisać:

```
10 N=1
20 :
30 INPUT "Ocena? ";OCN(N)
40 N=N+1
50 IF N>5 THEN 100
60 GOTO 30
70 :

100 STOP
```

Wyprowadzanie tablicy OCN( )

Wypełnijmy tablicę OCN( ) i wyprowadźmy następnie jej zawartość.

```
10 FOR N=1 TO 4
20  INPUT "Ocena? ";OCN(N)
30 NEXT N
40 REM -------- Wyprowadzanie tablicy OCN()
50 PRINT
60 PRINT "Lista ocen"
70 PRINT
80 FOR N=1 TO 4
90   PRINT "Ocena";N;OCN(N)
100 NEXT N

RUN↵

Ocena? 12↵
Ocena? 16↵
Ocena? 13↵
Ocena? 11↵

Lista ocen

Ocena1 12
Ocena2 16
Ocena3 13
Ocena4 11
```

Deklarowanie rozmiaru tablicy: DIM (liczba elementów)

Jeżeli tablica ma zawierać więcej niż 10 elementów, to musi być zadeklarowana maksymalna liczba elementów tablicy, ze względu na konieczność zarezerwowania dla nich miejsca w pamięci komputera.

Oceny mogą być definiowane za pomocą DATA:

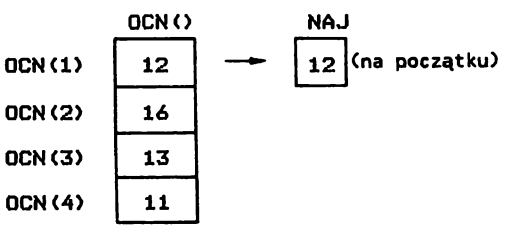
```
 10 DATA 12,16,13,11
 20 :
 25 DIM OCN(4)
 30 FOR N=1 TO 4        :REM czytanie danych DATA
 40  READ OCN(N)
 50 NEXT N
 60 REM ------ Sumowanie i wyznaczanie sredniej
 70 SOCEN=0
 90 FOR N=1 TO 4
100  SOCEN=SOCEN+OCN(N)
110 NEXT N
120 :
130 PRINT "Suma/Srednia:";SOCEN;SOCEN/4
```

RUN

```
Suma/Srednia: 52  13
```

Wyszukiwanie najlepszej oceny

Zakładamy wstępnie, że pierwsza ocena jest najlepsza. Następnie porównujemy ją z drugą oceną. Jeżeli druga ocena jest większa, to staje się ona najlepsza itd...

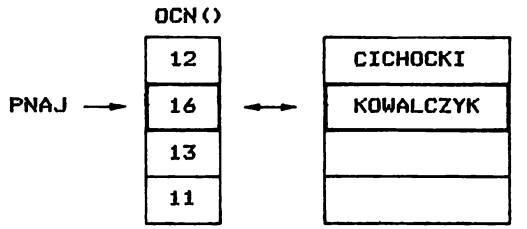



```
 10 DIM OCN(4)
 20 REM -------- Wprowadzanie ocen
 30 FOR N=1 TO 4
 40 INPUT "Ocena? ";OCN(N)
 50 NEXT N
 60 REM ------ Wyszukiwanie najlepszej oceny
 70 NAJ=OCN(1)
 80 FOR N=2 TO 4
 90  IF OCN(N)>NAJ THEN NAJ=OCN(N)
100 NEXT N
110 REM ------------
120 PRINT "Najlepsza ocena: ";NAJ
```

RUN

```
.
.
Najlepsza ocena: 16
```

W przedstawionym niżej przykładzie nie zapamiętujemy najlepszej oceny, lecz jej pozycję w tablicy.




```
10 DIM OCN(4)
20 DIM UCZ$(4)
30 REM ---------------------------------
40 FOR N=1 TO 4
50   INPUT "Ocena? ";OCN(N)
60   INPUT "Uczen? ";UCZ$(N)
70 NEXT N
80 REM ---------------------------------
90 PNAJ=1
100 FOR N=2 TO 4
110   IF OCN(N)>OCN(PNAJ) THEN PNAJ=N
120 NEXT N
130 REM ---------------------------------
140 PRINT "Najlepsza ocena: ";OCN(PNAJ),UCZ$(PNAJ)
RUN↵

Ocena? 12↵
Uczen? CICHOCKI↵
Ocena? 16↵
...
Najlepsza ocena: 16        KOWALCZYK

PRINT PNAJ↵
2
```

Tryb bezpośredni

Uwaga: Operacje, które wykonywaliśmy dotychczas (sumowanie, wyznaczanie średniej, wyszukiwanie najlepszej oceny) można było zrealizować bez użycia tablic. Jednakże w przypadku bardziej złożonych operacji, takich jak na przykład sortowanie, stosowanie tablic staje się niezbędne.

Sortowanie ocen

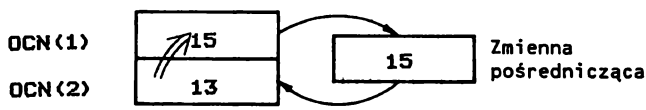
Posortujmy oceny zawarte w tablicy OCN( ).

Porównujemy najpierw drugi element tablicy z elementem pierwszym:

- Jeżeli jest on większy, to nie zmieniamy kolejności elementów.
- Jeżeli jest on mniejszy, to przestawiamy elementy w celu ich uporządkowania.

```
230 IF OCN(I+1)<OCN(I) THEN X=OCN(I):OCN(I)=OCN(I+1):OCN(I+1)=X:IW=1
    ↑                      ↑
JEZELI OCN(I+1)<OCN(I)  TO  przestawienie OCN(I+1) i OCN(I)
```

Przestawianie ocen OCN(I) i OCN(I + 1) jest realizowane za pomocą zmiennej pośredniczącej X:

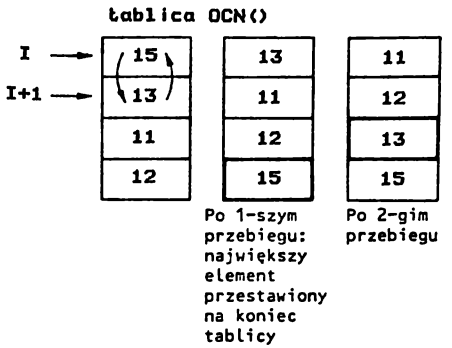


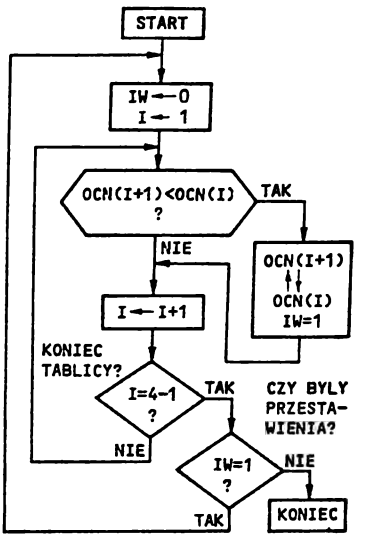
Porównujemy następnie w ten sam sposób element trzeci z drugim (zwiększając indeks I o 1) i przestawiamy elementy jeżeli nie są one uporządkowane. I tak dalej...

Po porównaniu wszystkich przylegających do siebie elementów tablicy, największy z elementów tablicy jest "przestawiony" na jej koniec. (Zwróć uwagę, że dla N elementów liczba wykonanych porównań wynosi N-1). W tym momencie sortowanie może, ale nie musi, być zakończone.

Aby to stwierdzić, wystarczy skontrolować wskaźnik przestawień (inwersji) IW. Jeżeli podczas badania tablicy nie przestawiano elementów, to znaczy że OCN(1) < OCN(2) < OCN(3) < OCN(4). Elementy są więc uporządkowane.

Jeżeli natomiast były wykonywane jakieś przestawienia, to badamy tablicę ponownie. Po zakończeniu drugiego badania, największy z N-1 elementów przechodzi na przedostatnią pozycję. W celu posortowania tablicy należy wykonać maksymalnie N badań.






```
40 FOR I=1 TO 4
50 INPUT "Ocena? ";OCN(I)
70 NEXT I
200 REM ------------ Sortowanie ocen ---------------
205 :
210 IW=0                    :REM Wskaznik inwersji
220 FOR I=1 TO 4-1
230  IF OCN(I+1)<OCN(I) THEN X=OCN(I):OCN(I)=OCN(I+1):OCN(I+1)=X:IW=1
240 NEXT I
250 IF IW=1 THEN 210        :REM Czy byly wykonywane inwersje?
255 :
260 REM ------ Wyprowadzanie posortowanych ocen -----
270 FOR I=1 TO 4
280  PRINT OCN(I)
290 NEXT I
```

```
RUN

Ocena? 15
Ocena? 13
Ocena? 11
Ocena? 12

11
12
13
15
```

Aby dokładniej prześledzić przebieg sortowania, możesz wstawić linię:

```
245 FOR K=1 TO 4:PRINT OCN(K):NEXT K:PRINT
```

Po każdym badaniu tablicy wyprowadzamy jej zawartość.

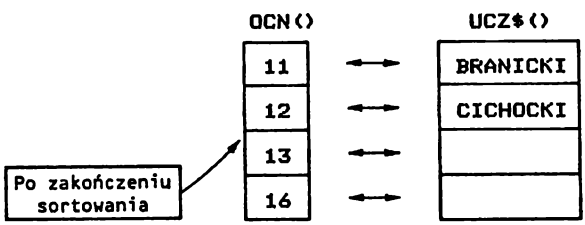
```
RUN
13
11
12
15
```

1-szy przebieg: największa wartość przestawiona na koniec tablicy

```
11
12
13
15
```

2-gi przebieg

Chcąc uzyskać listę ocen uporządkowanych według wzrastających wartości, wraz z listą nazwisk, wystarczy przestawiać elementy w tablicy UCZ$( ) za każdym razem, gdy przestawiamy elementy w tablicy OCN( ).




```
200 REM ------------------------------ Sortowanie
210 IW=0                    :REM wskaznik inwersJi
220 FOR I=1 TO 4-1
230 IF OCN(I+1)<OCN(I) THEN X=OCN(I):OCN(I)=OCN(I+1):OCN(I+1)=X:
                X$=UCZ$(I):UCZ$(I)=UCZ$(I+1):UCZ$(I+1)=X$:IW=1
240 NEXT I
250 IF IW<>0 THEN 210
260 REM ------------------- Wyprowadzanie wynikow
270 FOR I=1 TO 4
280  PRINT OCN(I),UCZ$(I)
290 NEXT I

RUN

11          BRANICKI
12          CICHOCKI
13          BARANOWSKI
16          KOWALCZYK
```

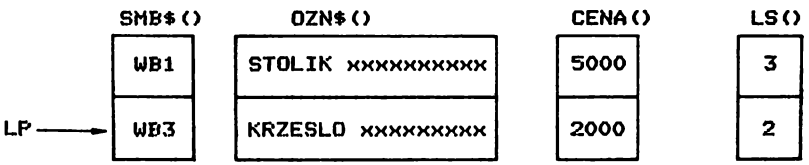
Aby uzyskać listę posortowaną według alfabetycznego porządku nazwisk, wystarczy napisać:

```
230 IF UCZ$(I+1)<UCZ$(I) THEN ......
```

Wystawianie faktury

Poszczególne wyroby są opisane (przez podanie symbolu, nazwy i ceny) w danych DATA.

Opisy wyrobów, na które ma być wystawiona faktura, są gromadzone w czterech tablicach: SYMB$(), OZN$(), CENA() i LS(). Po uwzględnieniu wszystkich wyrobów jest wyprowadzana faktura.



Wykonanie programu:

```
Nazwisko klienta? JERZY PAWLICKI ↵
Adres?            01493 Warszawa xxxxxx ↵
Symbol wyrobu? (K - koniec) WB1 ↵
Sztuk?                      3 ↵
Symbol wyrobu? (K - koniec) WB3 ↵
Sztuk?                      2 ↵
Symbol wyrobu? (K - koniec) K ↵

Wystawianie faktury xxxxx

                          Faktura dla:JERZY PAWLICKI
                                      01493 Warszawa xxxxxx

WB1  STOLIK xxxxxxxxxx      5000      3          15000
WB3  KRZESLO xxxxxxxxx      2000      2          4000

                                      W sumie:   19000
```

```
10 REM FT      FAKTURA
20 :
30 REM ---------- Symbole, oznaczenia, ceny wyrobow
40 :
50 DATA WB1,"STOLIK xxxxxxxxx",5000
60 DATA WB2,"FOTEL xxxxxxxxxx",7500
70 DATA WB3,"KRZESLO xxxxxxxxx",2000
80 DATA *
90 REM ----------- Gromadzenie danych
100 INPUT "Nazwisko klienta? ";NAZW$
110 INPUT "Adres?            ";ADR$
120 :
130 LP=0          :REM liczba pozycji
140 :
150 INPUT "Symbol wyrobu? (K - koniec) ";S$
160 IF S$="K" THEN 330
170 INPUT "Sztuk?                      ";LS
180 :
190 RESTORE       :REM ustawienie na poczatek DATA
200 :
210 READ SMB$:IF SMB$="*" THEN PRINT "Brak wyrobu":GOTO 150
220 READ OZN$
230 READ CENA
240 IF S$<>SMB$ THEN 210
250 LP=LP+1
260 SMB$(LP)=SMB$:OZN$(LP)=OZN$
270 CENA(LP)=CENA:LS(LP)=LS
300 :
310 GOTO 150
320 REM --------------- Wyprowadzanie faktury
330 PRINT
340 PRINT "Wystawianie faktury xxxxx":PRINT
350 PRINT TAB(28);"Faktura dla:";TAB(40);NAZW$
360 PRINT TAB(40);ADR$
370 PRINT
380 S=0
390 FOR P=1 TO LP
400   PRINT SMB$(P);
410   PRINT TAB(6);OZN$(P);
420   PRINT TAB(30);CENA(P);
430   PRINT TAB(40);LS(P);
440   PRINT TAB(50);LS(P)*CENA(P)
450   S=S+LS(P)*CENA(P)
460 NEXT P
470 PRINT
480 PRINT TAB(40);"W sumie:";TAB(50);S
```

Układanie menu

Możesz wybrać jedną z trzech zakąsek, jedno z trzech dań głównych i jeden spośród trzech deserów. Dysponujesz kwotą 280 złotych.

Obliczane są wszystkie kombinacje zakąsek, dań głównych i deserów. Pod uwagę brane są tylko te z nich, których łączna wartość nie przekracza 280 złotych.

Gdybyśmy chcieli drukować nazwy zakąsek, dań głównych i deserów, należałoby zdefiniować je za pomocą DATA i wczytać do trzech tablic (analogicznie jak w przypadku cen).

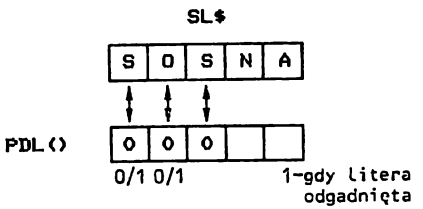
```
10 REM MENU       Układanie menu
20 :
30 BUDZ=250       :REM Kwota do dyspozycji
40 :
50 DATA 40,50,120
60 DATA 150,180,250
70 DATA 20,70,100
80 :
90 FOR Z=1 TO 3:READ CZ(Z):NEXT Z
100 FOR P=1 TO 3:READ CP(P):NEXT P
110 FOR D=1 TO 3:READ CD(D):NEXT D
120 :
130 FOR Z=1 TO 3
140  FOR P=1 TO 3
150   FOR D=1 TO 3
160     S=CZ(Z)+CP(P)+CD(D)
170     IF S>BUDZ THEN 190
180     LPRINT "Zakaska:";Z;"Danie:";P;"Deser:";D;"Suma:";S
190   NEXT D
200  NEXT P
210 NEXT Z
```

CZ()	CP()	CD()
40	150	20
50	180	70
120	250	100
Ceny zakąsek	Ceny dań głównych	Ceny deserów

```
Zakaska: 1 Danie: 1 Deser: 1 Suma: 210
Zakaska: 1 Danie: 1 Deser: 2 Suma: 260
Zakaska: 1 Danie: 2 Deser: 1 Suma: 240
Zakaska: 2 Danie: 1 Deser: 1 Suma: 220
Zakaska: 2 Danie: 1 Deser: 2 Suma: 270
Zakaska: 2 Danie: 2 Deser: 1 Suma: 250
```

Odgadywanie słów

Program wybiera jedno ze słów zdefiniowanych w danych DATA. Gra polega na odgadnięciu tego słowa.
Gracz podaje litery, a program wskazuje pozycję lub pozycje tych liter w poszukiwanym słowie.




```
10 REM ZGAD      Odgadywanie slow
20 CLEAR 1000
30 DATA SOSNA,DOM,TACZKA,GRUSZKA,SZAFKA
40 DATA *
50 :
60 DIM PDL(15)              :REM Tablica odgadnietych liter
70 :
80 READ SL$:IF SL$="*" THEN RESTORE:GOTO 80
90 L=LEN(SL$)
100 FOR P=1 TO 15:PDL(P)=0:NEXT P
110 PRINT:PRINT "Odgadnij slowo na ";L;"liter":PRINT
120 :
130 PRINT:INPUT "Podaj litere: ";L$
140 :
150 FOR P=1 TO L            :REM Poszukiwanie podanej litery w slowie
160  IF L$=MID$(SL$,P,1) THEN PDL(P)=1
170 NEXT P
180 REM ------------------- Wyswietlanie wyniku
190 LDL=0
200 FOR P=1 TO L
210  IF PDL(P)=1 THEN PRINT MID$(SL$,P,1);:LDL=LDL+1:GOTO 230
220  PRINT ".";
230 NEXT P
240 IF LDL<L THEN 130
250 PRINT:PRINT "Dobrze"
260 GOTO 70
270 '
```

```
RUN↵

Odgadnij slowo na 5 liter

Podaj litere: N↵
...N.
Podaj litere: S↵
S.SN.
```

Ulepszenie programu

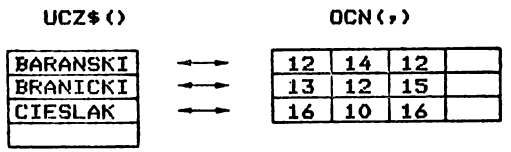
Słowa podawane są w takiej kolejności, w jakiej zostały zdefiniowane w DATA. Można jednak spowodować, aby były one wybierane losowo, pisząc:

```
25 N=5                                  :REM N - liczba slow
80 RESTORE
82 X=RND(1)*N                           :REM RND(N) na TRS-80
84 FOR I=1 TO X:READ SL$:NEXT I         :REM przewiniecie o X slow
86 READ SL$
```

Tablica dwuwymiarowa

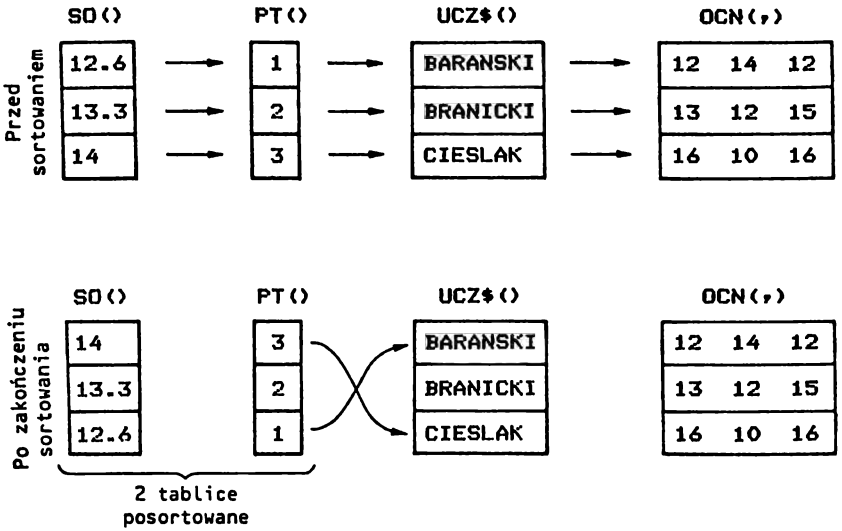
Tablice mogą mieć wiele wymiarów (do 255)[1].
Załóżmy, że chcemy wyprowadzić oceny uczniów zdefiniowane w DATA. Chcąc uzyskać listę tych ocen moglibyśmy odczytywać dane i wyprowadzać je w niezmienionej kolejności.

Użycie tablicy dwuwymiarowej umożliwia nam wykonanie dodatkowych operacji (takich jak na przykład sortowanie). Każdy wiersz tablicy zawiera oceny jednego ucznia.



W celu uzyskania listy uczniów i ich ocen w kolejności malejącej średniej, skorzystamy z tablicy pośredniczącej PT( ). Tablica PT( ) 'wskazuje' wiersze tablicy ocen OCN(,). Sortowaniu podlega tablica średnich ocen SO( ), przy czym za każdym razem, gdy przestawiane są dwa elementy tablicy SO( ), przedstawiane są również odpowiednie elementy tablicy PT( ). Dzięki użyciu tablicy PT( ) nie musimy przestawiać ocen w tablicy OCN(,), co byłoby dość uciążliwe.

[1] Jeżeli liczba elementów w którymkolwiek wymiarze tablicy jest większa niż 10, to użycie tablicy w programie musi być poprzedzone deklaracją DIM w postaci:
DIM (liczba elementów 1, liczba elementów 2, ...) (przyp. tłum.).



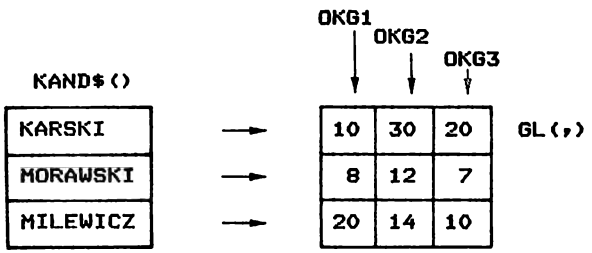

```
10 REM AO       ANALIZA OCEN
20 :
30 DATA "BARANSKI",12,14,12,99
40 DATA "BRANICKI",13,12,15,99
50 DATA "CIESLAK", 16,10,16,99
60 DATA *
70 REM ---------------- Czytanie danych DATA
80 DIM UCZ$(30),SO(30),PT(30)
90 DIM OCN(30,10)
100 :
110 FOR U=1 TO 30
120  READ X$:IF X$="*" THEN LU=U-1:GOTO 210
130  UCZ$(U)=X$
140 :
150  FOR N=1 TO 10
160    READ X:IF X=99 THEN 190
170    OCN(U,N)= X
180  NEXT N
190 NEXT U
200 REM --------------- Drukowanie tablic UCZ$() i OCN(,)
210 FOR U=1 TO LU
220  LPRINT UCZ$(U);TAB(15);
230  FOR N=1 TO 10
240   IF OCN(U,N)=0 THEN 270
```

```
250   LPRINT TAB(15+3*N);OCN(U,N);
260  NEXT N
270  LPRINT
280 NEXT U
290 REM --------- Obliczanie srednich ocen - tablica SO()
300 LPRINT:LPRINT "Srednie oceny":LPRINT
310 FOR U=1 TO LU
320  SM=0
330  FOR N=1 TO 10
340    IF OCN(U,N)=0 THEN LO=N-1:GOTO 370
350    SM=SM+OCN(U,N)
360  NEXT N
370  SO(U)=SM/LO
380  LPRINT "Sr:";UCZ$(U);TAB(15);SO(U)
390 NEXT U
400 REM ========== Drukowanie listy w kolejnosci srednich ocen
410 FOR I=1 TO 30:PT(I)=I:NEXT I
420 REM ---------------------------- Sortowanie
430 IW=0
440 FOR I=1 TO LU-1
450   IF SO(I)>SO(I+1) THEN X=SO(I):SO(I)=SO(I+1):SO(I+1)=X:
                           X=PT(I):PT(I)=PT(I+1):PT(I+1)=X:IW=1
460 NEXT I
470 IF IW=1 THEN 430
480 REM ---------------------- Drukowanie
485 LPRINT:LPRINT "Lista w kolejnosci srednich ocen":LPRINT
490 FOR U=1 TO LU
500   LPRINT UCZ$(PT(U));TAB(15);
510   FOR N=1 TO 10
520    IF OCN(PT(U),N)=0 THEN 550
530    LPRINT TAB(15+3*N);OCN(PT(U),N);
540   NEXT N
550   LPRINT TAB(40);SO(U)
560 NEXT U
```

```
BARANSKI        12  14  12
BRANICKI        13  12  15
CIESLAK         16  10  16

Srednie oceny

Sr:BARANSKI     12.6667
Sr:BRANICKI     13.3333
Sr:CIESLAK      14


Lista w kolejnosci srednich ocen
CIESLAK         16  10  16          14
BRANICKI        13  12  15          13.3333
BARANSKI        12  14  12          12.6667
```

Wybory

Zarządzaniem jest objętych kilka okręgów wyborczych. Wyniki dla każdego kandydata i każdego okręgu wyborczego są gromadzone w dwuwymiarowej tablicy GL(,).

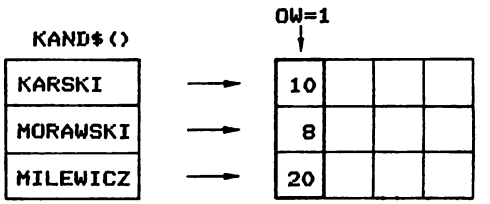


W pierwszym trybie (KAND) są wprowadzane nazwiska kandydatów i jest wypełniana tablica KAND$().

```
Tryb? ( KAND,OKGW,WYNIKI ) KAND↵
Nazwisko kandydata? (K-koniec) KARSKI↵
Nazwisko kandydata? (K-koniec) MORAWSKI↵
Nazwisko kandydata? (K-koniec) MILEWICZ↵
Nazwisko kandydata? (K-koniec) K↵
```

W drugim trybie (OKGW) jest wypełniana tablica GL(,) dla określonego okręgu wyborczego (1,2,...). W przedstawianym przykładzie nie ma możliwości kumulowania głosów w trakcie ich zliczania; należy podawać sumaryczną liczbę głosów oddanych na kandydata w danym okręgu. Poprzednia wartość może być jednak zmieniana.

```
Tryb? ( KAND,OKGW,WYNIKI ) OKGW↵

Okres? (0-koniec) 1↵

KARSKI          0    Liczba glosow? (0-bez zmian) 10↵
MORAWSKI        0    Liczba glosow? (0-bez zmian) 8↵
MILEWICZ        0    Liczba glosow? (0-bez zmian) 20↵
```

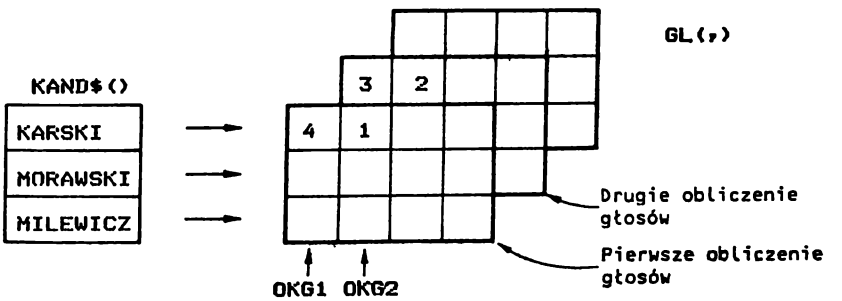


W trybie trzecim (WYNIKI) są wyprowadzane wyniki: suma głosów oddanych na każdego kandydata oraz udział procentowy tych głosów w łącznej liczbie głosów.

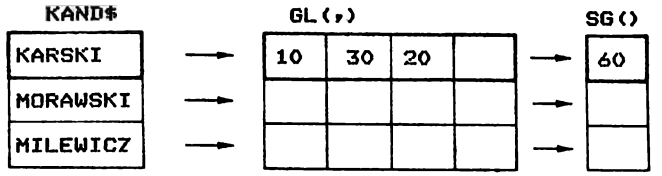
Kumulowanie głosów w trakcie ich zliczania w okręgu wyborczym można zrealizować dwoma sposobami:

- Stosując tę samą dwuwymiarową tablicę GL(,), dodając głosy w miarę ich napływania.
- Definiując trójwymiarową tablicę GL(,,) i przechowując w niej wyniki wszystkich obliczeń pośrednich.

Drugie rozwiązanie jest godne polecenia ze względu na możliwość uzyskania informacji o przebiegu głosowania.



W celu uzyskania listy wyników uporządkowanej według liczby głosów oddanych na poszczególnych kandydatów, należy utworzyć tablicę SG( ) i posortować ją.



a) sumowanie głosów w tablicy SG( ):

```
600 FOR K=1 TO LK
610   SG(K)=0
620   FOR OW=1 TO 5
630     SG(K)=SG(K)+GL(K,OW)
640   NEXT OW
650 NEXT K
```

b) sortowanie tablic SG( ) i KAND$( ):

```
700 IW=0
710 FOR I=1 TO LK-1
720    IF SG(I+1)>SG(I) THEN X=SG(I):SG(I)=SG(I+1):SG(I+1)=X:
          X$=KAND$(I):KAND$(I)=KAND$(I+1):KAND$(I+1)=X$:IW=1
730 NEXT I
740 IF IW=1 THEN 700
```

c) wyprowadzanie tablic SG( ) i KAND$( ):

```
800 FOR K=1 TO LK
810    PRINT SG(K);TAB(6);KAND$(K)
820 NEXT K

60    KARSKI
44    MILEWICZ
27    MORAWSKI
```

Uwaga: Adaptując program dla APPLE lub TRS-80 należy zastąpić nazwę KAND$ przez KND$ (ponieważ AND jest słowem kluczowym języka Basic).

```
10 REM WB           WYBORY
20 :
30 REM     10 kandydatow / 5 okresow wyborczych
40 :
50 DIM KAND$(10),GL(10,5)
60 REM ---------------------- Wybor trybu ------------
70 :
80 PRINT:INPUT "Tryb? ( KAND,OKGW,WYNIKI ) ";TR$
90 IF TR$="KAND" THEN GOSUB 140
100 IF TR$="OKGW" THEN GOSUB 240
110 IF TR$="WYNIKI" THEN GOSUB 350
120 GOTO 80
130 REM --------------------- Nazwiska kandydatow ----
140 PRINT:INPUT "Nazwisko kandydata? (K-koniec) ";K$:IF K$="K" THEN 220
150 :
160 FOR K=1 TO 10
170  IF K$=KAND$(K) THEN PRINT:PRINT "Powtorzone nazwisko":GOTO 140
180  IF KAND$(K)="" THEN KAND$(K)=K$:LK=K:GOTO 140
190 NEXT K
200 PRINT "Za duzo kandydatow":STOP
210 :
220 RETURN
230 REM ------------------------- Glosy --------
240 PRINT:OW=0:INPUT "Okres? (0-koniec) ";OW:IF OW=0 THEN RETURN
250 :
260 FOR K=1 TO LK
270  PRINT KAND$(K);TAB(15);
280  PRINT GL(K,OW);TAB(20);
290  G=0:INPUT "Liczba glosow? (0-bez zmian) ";G:IF G=0 THEN 310
300  GL(K,OW)=G
310 NEXT K
320 RETURN
330 REM ----------------- Drukowanie wynikow -------
340 :
350 S=0                     :REM Suma glosow
360 FOR K= 1 TO LK
370  FOR OW=1 TO 5:S=S+GL(K,OW):NEXT OW
380 NEXT K
390 :
400 LPRINT:LPRINT "Wyniki":LPRINT
410 FOR K=1 TO NK
420  LPRINT KAND$(K);TAB(15);
430  SK=0
440  FOR OW=1 TO 5
450   SK=SK+GL(K,OW)
460   IF GL(K,OW)<>0 THEN LPRINT TAB(15+OW*4);GL(K,OW);
470  NEXT OW
480  LPRINT TAB(40);SK;
490  LPRINT TAB(45);"%";SK/S*100
500 NEXT K
510 RETURN
520 '
530 ' Wyniki
540 '
550 ' KARSKI          10  30  20          60   %45.8
560 ' MORAWSKI        8   12  7           27   %20.6
570 ' MILEWICZ        20  14  10          44   %33.5
```

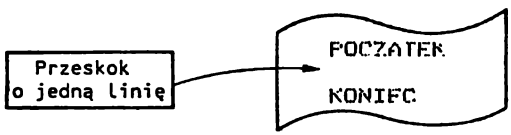
Wyprowadzanie wyników

Przedstawimy zwięźle najczęściej stosowane formy wyprowadzania wyników.

PRINT zmienna

PRINT, po którym występuje zmienna lub stała, powoduje wyprowadzenie tej zmiennej lub stałej, a następnie przeskok do nowej linii i przejście do jej początku.
Samo PRINT powoduje przejście do początku nowej linii.

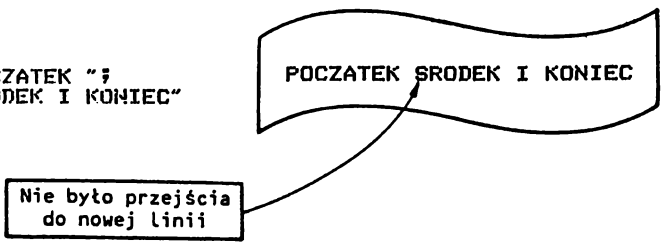
```
10 PRINT "POCZATEK"
20 PRINT
30 PRINT "KONIEC"
```



PRINT zmienna ;

Znak ; (średnik) blokuje przeskok do nowej linii.

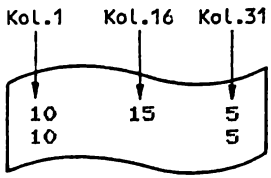
```
10 PRINT "POCZATEK ";
20 PRINT "SRODEK I KONIEC"
```



PRINT zmienna ,

Znak , (przecinek) powoduje standardową tabulację wyprowadzanych wartości w kolumnach 1, 16, 31, ...

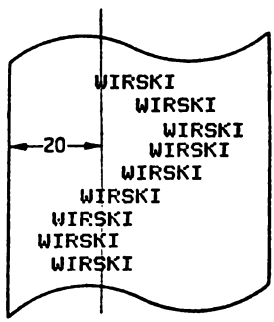
```
10 X=10:Y=15:Z=5
20 PRINT X,Y,Z
30 PRINT X, ,Z
```



PRINT TAB (pozycja)

Przesuwa początek drukowania (lub kursor) do określonej kolumny bieżącej linii, co umożliwia uzyskanie dowolnej tabulacji.

```
10 INPUT "Nazwisko? ";NAZW$
20 :
30 FOR K=0 TO 6.28*3 STEP 6.28/10
40  X=20+SIN(K)*10
50  PRINT TAB(X);NAZW$
60 NEXT K
```



Rysunek wypełnionego trójkąta.

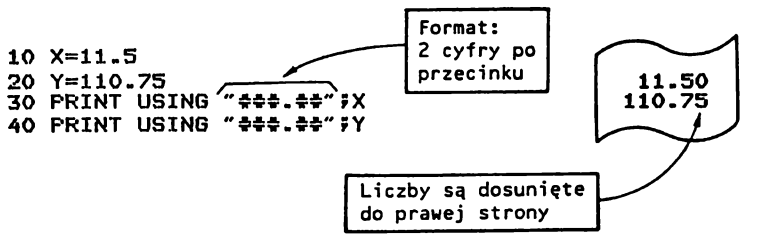
```
10 FOR L=1 TO 9 STEP 2
20  PRINT TAB(20-L/2);
30  FOR G=1 TO L
40    PRINT "*";
50  NEXT G
60 PRINT
70 NEXT L
```

```
*
***
*****
*******
*********
```

PRINT USING "# # #.# #"; zmienna

(Microsoft 5., TRS-80)

Umożliwia uzyskanie **prawostronnej** tabulacji wartości numerycznych.
('PRINT zmienna' tabuluje z **dosunięciem do lewej strony**).
W formacie # # #.# # liczba znaków # po kropce określa liczbę cyfr części ułamkowej, które mają być wyprowadzone (w tym przykładzie 2).

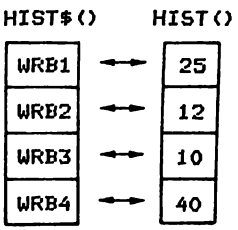


Tabulację prawostronną można uzyskać także nie stosując PRINT USING:

```
10 DEF FNC(X)=7-LEN(STR$(INT(X)))
20 :
70 A=13
80 B=131
100 PRINT TAB(FNC(A));A
110 PRINT TAB(FNC(B));B

RUN

    13
   131
```

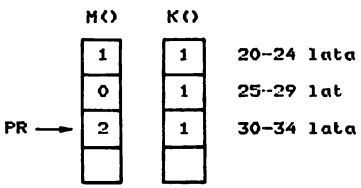
Histogramy

```
10 'HGM       HISTOGRAM
20 '
30 '     Program drukuje histogram sprzedazy wyrobow.
40 '     Wysokosc kratek jest proporcjonalna do udzialu
50 '     procentowego danego wyrobu w sprzedazy ogolnej.
60 '
70 N=4
80 DATA WRB1,25,WRB2,12,WRB3,10,WRB4,40
90 FOR I=1 TO N
100  READ HIST$(I)
110  READ HIST(I)
120 NEXT I
130 '------------- Skala
140 SM=0
150 FOR I=1 TO N
160  SM=SM+HIST(I)
170 NEXT I
180 '
190 SKL=20/SM
200 '-------------------------  Drukowanie
210 LPRINT "---------------"
220 FOR I=1 TO N
230   LPRINT "!";TAB(1);HIST(I);TAB(5);HIST$(I);TAB(12);
240   LPRINT "%";INT((HIST(I)/SM*100)+.5);:LPRINT TAB(17);"!"
250   FOR K=1 TO HIST(I)*SKL-2
260     IF HIST(I)*SKL-1<1 THEN 280
270     LPRINT "!                  !"
280   NEXT K
290   LPRINT "-------------------"
300 NEXT I
310 '-----------------
320 LPRINT:LPRINT "Suma:";SM
```




```
-------------------
! 25 WRB1   % 29 !
!                 !
!                 !
!                 !
-------------------
! 12 WRB2   % 14 !
!                 !
-------------------
! 10 WRB3   % 11 !
!                 !
-------------------
! 40 WRB4   % 46 !
!                 !
!                 !
!                 !
!                 !
!                 !
!                 !
!                 !
-------------------

Suma: 87
```

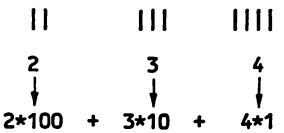
```
10 REM          HISTOGRAM
20 :
30 REM     Rozklad ws przedzialow wieku
40 :
50 DATA 43,M,34,K,23,M,56,K,36,K,57,M
60 DATA 21,K,32,M,55,K,34,M,35,M,27,K
70 DATA 999
80 :
90 READ WIEK:IF WIEK=999 THEN 190
100 READ SX$
110 IF WIEK<20 OR WIEK>59 THEN 90
120 :
130 PR=INT(WIEK-20)/5+1
140 IF SX$="M" THEN M(PR)=M(PR)+1
150 IF SX$="K" THEN K(PR)=K(PR)+1
160 :
170 GOTO 90
180 REM ------------ Skala
190 MX=0
200 FOR PR=1 TO 8
210   IF M(PR)+K(PR)>MX THEN MX=M(PR)+K(PR)
220 NEXT PR
230 SKL=10/MX
240 :
250 REM ---------------------- Drukowanie ---------------
260 :
270 LPRINT "Wiek";TAB(15);"Razem";TAB(28);"Mezczyzni";TAB(40);"Kobiety"
275 LPRINT
280 FOR PR=1 TO 8
290   LPRINT 20+5*(PR-1);"-";
300   LPRINT 20+5*PR-1;
310   LPRINT M(PR)+K(PR);TAB(15);
320 :
330   X=(M(PR)+K(PR))*SKL
340   FOR I=1 TO X
350     IF X<1 THEN 370
360     LPRINT "*";
370   NEXT I
380 :
390   LPRINT TAB(28);
400   X=M(PR)*SKL
410   FOR I=1 TO X
420     IF X<1 THEN 440
430     LPRINT "M";
440   NEXT I
450 :
460   LPRINT TAB(40);
470   X=K(PR)*SKL
480   FOR I=1 TO X
490     IF X<1 THEN 510
500     LPRINT "K";
510   NEXT I
520   LPRINT
530 NEXT PR
```



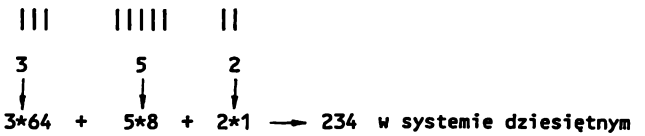

```
Wiek          Razem          Mezczyzni    Kobiety
 20 - 24  2   ******         MMM          KKK
 25 - 29  1   ***                         KKK
 30 - 34  3   *********      MMMMMM       KKK
 35 - 39  2   ******         MMM          KKK
 40 - 44  1   ***
 45 - 49  0
 50 - 54  0
 55 - 59  3   *********      MMM          KKKKKK
```

Systemy liczbowe

Aby przedstawić liczbę 234, moglibyśmy narysować 234 patyczki. Znacznie prościej jest jednak powiedzieć, że mamy:
4 jednostki
3 dziesiątki (3 wiązki po 10 patyczków)
2 setki (2 wiązki po 100 patyczków)



10 jest tutaj 'podstawą'. Równie dobrze możemy użyć podstawy 8:



Uogólniając, w systemie o podstawie P liczbę N przedstawiamy następująco:

$$N = a_n * P^n + a_{n-1} * P^{n-1} + \ldots + a_1 * P^1 + a_0 * P^0$$

1101 w systemie o podstawie 2 jest równe:

$$1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 \longrightarrow 1*8 + 1*4 + 0*2 + 1 \longrightarrow 13$$

```
10 REM KONWD   Konwersja (system o podstawie P --> system dziesietny)
20 :
30 INPUT "Podstawa? ";P
40 :
50 INPUT "Liczba? ";N
60 ND=0
70 :
80 FOR K=0 TO 6
90   Q=INT(N/10)            :REM Iloraz
100  R=N-(Q*10)            :REM Reszta: wspolczynnik k-tego rzedu
110 :
120  ND=ND+(P^K)*R
130  N=Q
140 NEXT K
150 :
160 PRINT ND
170 GOTO 50
```

N | 1 | 0 | 1 |

1	Reszta z dzielenia przez 10

```
RUN
Podstawa? 2
Liczba? 101
5
Liczba? 110
6
```

Zamiana liczb systemu dziesiętnego na liczby systemu o podstawie 2

Przedstawmy liczbę N w innej postaci.

$$N = a_n*P^n + a_{n-1}*P^{n-1} + \ldots + a_0*P^0 = \underbrace{(((a_n*P)+a_{n-1}*P+\ldots+a_2)*P+a_1)}_{\text{Iloraz}}*P+\underbrace{a_0}_{\text{Reszta}}$$

Dzieląc N przez P otrzymujemy iloraz

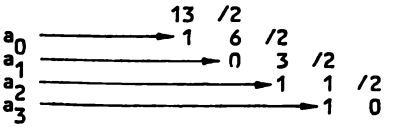
$$Q = (((a_n*P) + a_{n-1})*P + \ldots + a_2)*P + a_1$$

oraz resztę równą a_0.

Dzieląc ten iloraz przez P otrzymujemy resztę równą a_1.
W ten sposób reszty otrzymywane w wyniku kolejnych operacji dzielenia przez podstawę P wyznaczają współczynniki $a_0, a_1, \ldots, a_n$.

Przykład

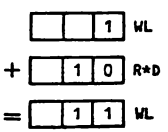
Zamieńmy liczbę 13 (w systemie dziesiętnym) na liczbę w systemie o podstawie 2.



1-sze rozwiązanie:

Zastosujemy bezpośrednio podaną wyżej metodę. Ponieważ kolejność, w jakiej powinny być wyprowadzane współczynniki, jest odwrotna niż kolejność ich wyznaczania, nie mogą być one wyprowadzane w trakcie przeprowadzania obliczeń. Gromadzimy je w zmiennej WL.

```
10 REM KONWP   Konwersja (system dziesietny --> system o podstawie P)
20 :
30 INPUT "Podstawa? "¦P
40 :
50 INPUT "Liczba? "¦N
60 :
70 WL=0:D=1                    :REM wynik do wyswietlenia - wasa dziesietna
80 :
90 Q=INT(N/P)                  :REM Iloraz
100 R=N-(Q*P)                  :REM Reszta
110 :
120 WL=R*D+WL
130 D=D*10
140 :
150 N=Q
160 IF Q>0 THEN 90
170 PRINT WL
180 GOTO 50

RUN ↵
Podstawa? 2↵
Liczba? 5↵
101
Liczba?
```



2-gie rozwiązanie:

Rozpoczynamy od wyznaczenia współczynnika najwyższego rzędu. Przy obliczeniach "ręcznych", metoda ta byłaby bardzo uciążliwa.

```
30 INPUT "Podstawa? ";P
40 :
50 INPUT "Liczba? ";N
60 :
70 FOR K=6 TO 0 STEP -1
80   X=P^K
90   Q=INT(N/X)
100  N=N-X*Q
110  PRINT Q;
120 NEXT K
```

RUN↵

```
Podstawa? 2↵
Liczba? 5↵
000101
```

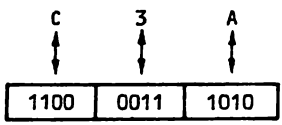
System szesnastkowy (heksadecymalny)

W systemie o **podstawie 16** należy zdefiniować symbole dla wartości 10, 11, 12, 13, 14, 15.

Wybierzmy symbol A dla oznaczenia wartości 10, B dla oznaczenia 11, itd.

Dlaczego system szesnastkowy jest wykorzystywany w informatyce?

System ten umożliwia przedstawianie liczb dwójkowych (binarnych) w krótszej postaci; każdej cyfrze systemu szesnastkowego odpowiadają 4 cyfry systemu dwójkowego:




```
 10 A$= 0123456789ABCDEF"
110 PRINT MID$(A$,Q+1,1);
```

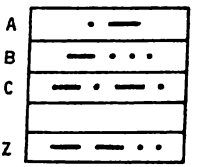
Możemy także napisać:

```
  5 DIM S$(16)
 10 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
 20 FOR I=1 TO 16:.READ S$(I): NEXT I
110 PRINT S$(Q+1);
```

Alfabet Morse'a

Za pomocą tego programu nauczysz się alfabetu Morse'a.
Przykładowo, po naciśnięciu litery S usłyszysz 3 krótkie dźwięki.
Po naciśnięciu litery O usłyszysz 3 długie dźwięki.

```
10 REM MORSE     Program do nauki alfabetu MORSE'a
20 :
80  DATA .-,-...,-.-.,-..,.,..-.
90  DATA --.,....,..,.---,-.-,.-..,--
100 DATA -.,---,.--.,--.-,.-.,...,-,..-
110 DATA ...-,.--,-..-,-.--,--..
120 :
130 DIM MRS$(26)
140 FOR P=1 TO 26
150  READ MRS$(P)
160 NEXT P
170 :
180 GET X$       :REM albo X$=INKEY$:IF X$="" THEN 180
190 :
200 P=ASC(X$)-64:IF P<1 OR P>26 THEN 180
210 :
220 X$=MRS$(P)                          Krótki dźwięk
230 FOR P=1 TO LEN(X$)
240  IF MID$(X$,P,1)="." THEN PRINT CHR$(7);
250  IF MID$(X$,P,1)="-" THEN FOR I=1 TO 30:PRINT CHR$(7);:NEXT I
260  FOR TM=1 TO 90:NEXT TM
270 NEXT P                              Długi dźwięk
280 GOTO 180
290 :
```



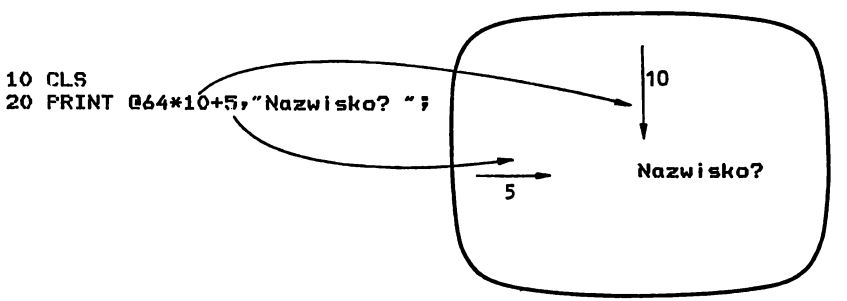
Bezpośrednie adresowanie ekranu

Bezpośrednie adresowanie ekranu umożliwia wyświetlenie tekstu w określonym miejscu ekranu (bez oddziaływania na pozostałą część ekranu).

TRS-80

PRINT@ 64*nr linii + nr kolumny, "tekst";

(64 jest liczbą znaków w jednej linii ekranu komputera TRS-80)

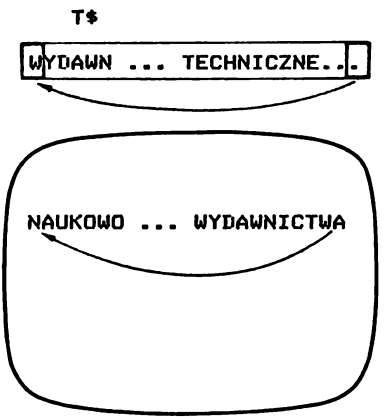


Poniżej podajemy dwa przykłady zastosowania tej instrukcji.

Gazeta świetlna

Program powoduje przesuwanie się tekstu na ekranie. Szybkość przesuwu jest uzależniona od wybranego opóźnienia (60 FOR TM = 1 TO 20: NEXT TM)

```
10 CLEAR 200
20 CLS
30 T$="WYDAWNICTWA NAUKOWO-TECHNICZNE..."
40 REM ----------------
50 PRINT @64*10+10,T$;
60 FOR TM=1 TO 20:NEXT TM    :REM Opoznienie
70 L=LEN(T$):T$=RIGHT$(T$,1)+LEFT$(T$,L-1)
80 GOTO 50
```

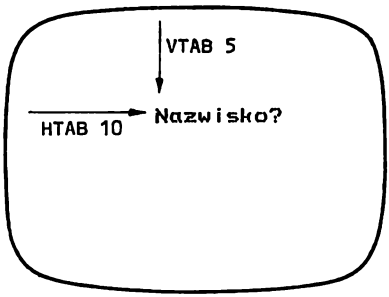


Migotanie tekstu

```
10 CLS
20 PRINT @64*10+10,"KUKU";
30 GOSUB 100
40 PRINT @64*10+10,"     ";
50 GOSUB 100
60 GOTO 20
70 :
100 FOR TM=1 TO 100:NEXT TM:RETURN    :REM Opoznienie
```

APPLE II

VTAB nr linii: HTAB nr kolumny




```
5 HOME                  :REM kasowanie ekranu
10 VTAB 5               :REM ustawienie kursora w linii 5
20 HTAB 10              :REM ustawienie kursora w kol.10
30 PRINT "Nazwisko? ";
```

Program *gazety świetlnej*, zaadaptowany dla APPLE II, ma następującą postać:

```
20 HOME
30 L$="WYDAWNICTWA NAUKOWO-TECHNICZNE..."
40 HTAB 5:VTAB 5:PRINT L$;
50 FOR TM=1 TO 40:NEXT TM
60 D=LEN(L$):L$=RIGHT$(L$,1)+LEFT$(L$,L-1)
70 GOTO 40
```

Przechwytywanie znaku z klawiatury

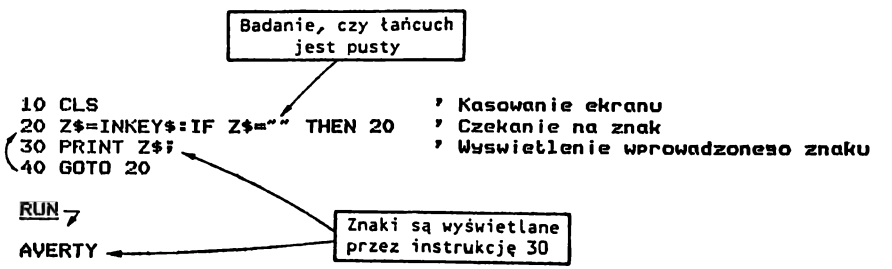
Instrukcja INPUT nie pozwala programowi na badanie znaków w trakcie ich wprowadzania, lecz dopiero po zatwierdzeniu ich przez operatora klawiszem <RETURN> albo <ENTER>.

Instrukcje **INKEY$, GET** *i* **INPUT$ (1)** *umożliwiają wykonywanie takiego badania.*

INKEY$ (TRS-80/Microsoft 5.)

Odczytuje **nieustannie** *znaki wprowadzane z klawiatury.*

Jeżeli nie został wprowadzony żaden znak, to odczytany tekst (w tym przykładzie Z$) jest pusty. Znak wprowadzony z klawiatury jest wyświetlany dopiero na żądanie programu (a nie "lokalnie" poprzez klawiaturę, jak w przypadku INPUT).



Poniższy program wyświetla znak o kodzie większym o 1 od kodu wprowadzonego znaku.

```
10 CLS
20 Z$=INKEY$:IF Z$="" THEN 20
30 PRINT CHR$(ASC(Z$)+1);
40 GOTO 20
```

```
RUN
Wprowadziłeś
AVERTY
BWFSUZ
```

Ten natomiast wyświetla jedynie znaki zawarte między 'A' i 'L'.

```
10 CLS
20 Z$=INKEY$:IF Z$="" THEN 20
30 IF Z$<"A" OR Z$>"L" THEN 20
40 PRINT Z$;
50 GOTO 20
```

```
RUN
Wprowadziłeś
AVERTY
AE
```

Pętla oczekiwania

Wyjście z pętli następuje po naciśnięciu przez operatora jakiegokolwiek klawisza.

```
10 X$=INKEY$:IF X$="" THEN 10    ' Czekanie
20 PRINT "Koniec czekania"
```

```
10 PRINT "Napisz S (szybko) "
20 N=0
30 :
40 O$=INKEY$:IF O$="S" THEN GOTO 100
50 N=N+1:IF N>1000 THEN PRINT "Za pozno":STOP
60 GOTO 40
70 :
100 ' dalszy ciag programu
```

GET znak (APPLE II).

Odczytuje znak wprowadzany z klawiatury.

Znak ten powinien być następnie wyświetlony przez program.

```
10 GET Z$
20 PRINT ASC(Z$);Z$
30 IF Z$="F" THEN 100
40 GOTO 10
50 :
100 END
```

Wyświetlanie wprowadzonego znaku

```
RUN
66 B
67 C
13
70 F
```

Kod litery B

Nacisnąłeś <RETURN> (w kodzie ASCII - 13)

A oto drugi przykład: *kompletowanie liczb.*

Program akceptuje jedynie cyfry. Jeżeli operator naciśnie na przykład literę, to nie zostanie ona wyświetlona, a ponadto nadany zostanie sygnał dźwiękowy.
Przyjmowane znaki są łączone w łańcuch L$.
Program zatrzymuje się po naciśnięciu przez operatora klawisza <RETURN> (kod 13).

Uwaga: Przedstawiony program nie reaguje prawidłowo na znak kasowania '←' (kod 8). Znak ten jest traktowany tak, jak wszystkie inne znaki.

```
10 L$=""
15 :
20 GET Z$
30 IF Z$<"0" OR Z$>"9" THEN PRINT CHR$(7);:GOTO 20
40 IF ASC(Z$)=13 THEN 100       :REM <RETURN>
50 PRINT Z$
60 L$=L$+Z$
70 GOTO 20
80 :
100 PRINT L$
110 PRINT VAL(L$)
```

L$ Z$

3	1	+	4

```
RUN
314
314
314
```

PRINT L$

PRINT VAL(L$)

Symulacja INKEY$ w komputerze APPLE

W odróżnieniu od INKEY$, *GET jest instrukcją* **blokującą**; Program oczekuje na wprowadzenie znaku i nie może wykonywać w tym czasie innych instrukcji.

W programach gier z ruchomymi figurami dla komputera APPLE II należy używać następującej sekwencji:

```
10 K=PEEK(-16384):IF K<128 THEN 10     :REM czekanie na znak
20 POKE -16368,0                       :REM zerowanie wskaznika klawiatury
30 PRINT CHR$(K-128);
40 GOTO 10
```

INPUT$ (1) (Microsoft 5.)

Instrukcja ta odczytuje znak wprowadzany z klawiatury. Tak samo jak GET w APPLE II, jest to instrukcja **blokująca.**

```
10 Z$=INPUT$(1)
20 PRINT Z$;
30 IF Z$="F" THEN 100
40 :
100 END
```

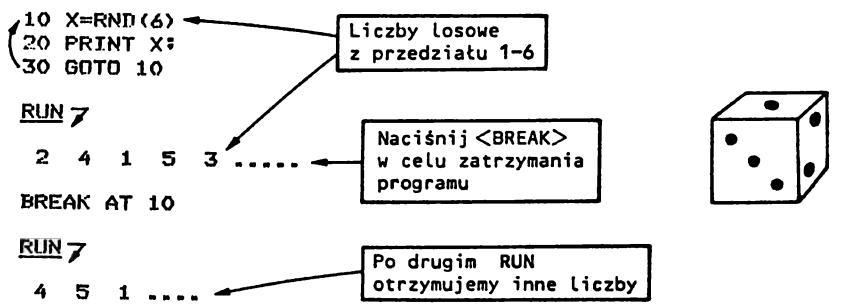
Liczby losowe

Liczby losowe stosuje się głównie w grach i w programach o charakterze edukacyjnym.
Składnia instrukcji jest różna w różnych wersjach języka Basic.
Przedstawimy instrukcje dla TRS-80, APPLE II oraz dla komputerów działających z systemem Basic Microsoft 5.

TRS-80

RND (X) dla $1 \leqslant X \leqslant 32767$

Generuje "przypadkową" liczbę całkowitą z przedziału 1—X (włącznie z X).



A oto przykład: **Sumowanie dwóch liczb**
Program wyświetla dwie liczby losowe z przedziału 1—100, żąda podania sumy tych liczb i sprawdza poprawność odpowiedzi.

```
10 A=RND(100)    :REM 1+INT(RND(1)*100) APPLE lub Microsoft 5.
20 B=RND(100)
30 '
40 PRINT "Jaka jest suma";A;"i";B;
50 INPUT "  Odpowiedz? ";ODP
60 IF ODP=A+B THEN PRINT "Dobrze":GOTO 10
70 PRINT "Nie ";A;"+";B;"=";A+B
80 GOTO 10

RUN↵

Jaka jest suma 25 i 48   Odpowiedz? 73↵
Dobrze
```

RND (X) dla $0 \leqslant X < 1$

Generuje liczbę losową z przedziału 0—1 (wyłącznie).

RANDOM

Inicjuje nowy ciąg losowy.

APPLE II

RND (X) dla X>0

Generuje liczbę losową z przedziału 0—1 (wyłączając 1)

```
10 PRINT RND(1)
20 GOTO 10

RUN↵

.212
.053
.67

BREAK AT 10

RUN↵

.042
.51
.234
```

Naciśnięcie <Ctrl C> w celu zatrzymania programu

Po drugim RUN otrzymujemy inny ciąg liczb

Aby uzyskać liczby całkowite z przedziału 0–9, należy:

- pomnożyć otrzymaną liczbę przez 10
- wydzielić część całkowitą wyniku za pomocą funkcji INT(X)

```
10 X=RND(1)
20 Y=X*10
30 Z=INT(Y)
40 PRINT X,Y,Z
50 GOTO 10
```

Wydzielenie części całkowitej Y
(PRINT INT(2.12) → 2)

```
RUN ↵

 .212        2.12        2
 .053        .53         0
 .67         6.7         6
```

RND(1) RND(1)*10 INT(RND(1)*10)

Aby uzyskać liczbę z przedziału 1–10, wystarczy dodać 1.

```
10 FOR N=1 TO 3
20  PRINT INT(RND(1)*10)+1;
30 NEXT N

RUN ↵

3 1 10
```

RND (X) dla X<0

Ujemna wartość X powoduje zainicjowanie ciągu losowego zależnego od wartości X. Dla określonej wartości X ciąg ten będzie zawsze taki sam.

```
10 X=RND(-2)
20 REM --------------
30 FOR I=1 TO 3
40  PRINT INT(RND(1)*6)+1;
50 NEXT I

RUN ↵

2 4 3 ...

RUN ↵

2 4 3 ...
```

Takie same ciągi

Gdyby w programie nie było linii '10 X = RND(−2)', wówczas ciąg losowy po drugim RUN byłby różny od ciągu uzyskanego po pierwszym RUN.

RND (0)

Podaje ostatnio wygenerowaną liczbę losową.

```
10 FOR I=1 TO 3
20  PRINT RND(1)
30 NEXT I

RUN ↵

 .567
 .033
 .12
PRINT RND(0) ↵
 .12
```

Tryb bezpośredni

Podaje ostatnią liczbę losową

Microsoft 5.

RND (X) dla X>0

Generuje liczbę losową z przedziału 0−1.

```
10 PRINT RND(1)
20 GOTO 10

RUN ↵

  .123
  .576
  .234

BREAK AT 10
```

RND (X) dla X<0

Inicjuje ciąg zależny od X.

```
10 X=RND(-3)
20 '
30 PRINT RND(1);
40 GOTO 30

RUN ↵
 .786   .023   .56  ...
RUN ↵
 .786   .023   .56  ...
```

Takie same ciągi (zależne od wartości X)

RND (0)

Podaje ostatnio wygenerowaną liczbę losową.

RANDOMIZE

Inicjuje ciąg losowy.

```
10 RANDOMIZE
20 FOR N=1 TO 3
30  PRINT RND(1);
40 NEXT N

RUN ↵

Random number seed? 5 ↵
 .423   .142   .673
```

Wprowadzana przez ciebie wartość

Ten sam efekt można uzyskać przez napisanie:

```
10 INPUT "Liczba? ";N
15 FOR I=1 TO N:X=RND(1):NEXT I
```

Dzielenie

Program wybiera losowo dwie liczby, wyświetla je na ekranie, oczekuje na podanie ilorazu tych liczb, a następnie sprowadza poprawność odpowiedzi.
Wstępnie jest ustalany stopień trudności (największa dzielna).

```
10 REM DZL     DZIELENIE
20 :
30 INPUT "Najwieksza dzielna? (do 1000)    ";DZ
40 :
50 X=INT(RND(1)*DZ)+1          :REM RND(DZ) dla TRS-80
60 Y=INT(RND(1)*DZ)/10         :REM RND(DZ/10) dla TRS-80
70 :
80 PRINT:PRINT "Jaki jest iloraz";X;"przez";Y;
90 INPUT "   Odpowiedz? ";ODP
100 IF INT(X/Y)=ODP THEN PRINT:PRINT "Dobrze":GOTO 50
110 :
120 PRINT:PRINT "Iloraz";X;"przez";Y;"wynosi";INT(X/Y)
130 GOTO 50
140 '
```

RUN ↵

Najwieksza dzielna? (do 1000) 100 ↵

Jaki jest iloraz 45 przez 6   Odpowiedz? 7 ↵

Dobrze

Grafika ...

Nieustannie zwiększają się możliwości komputerów w zakresie grafiki: z coraz większą precyzją definiuje się punkty, wprowadza się coraz efektywniejsze instrukcje.

Grafika komputera TRS-80[1] jest bardzo uboga: można jedynie zapalać lub gasić 128×48 punktów, bez możliwości wyboru koloru. Ze względu na swą prostotę TRS-80 ma jednak duże walory dydaktyczne.

W komputerze APPLE II istnieje natomiast możliwość bezpośredniego kreślenia prostych i wybierania koloru.

---

[1] Grafika polskiego mikrokomputera Meritum odpowiada grafice TRS-80 (przyp. tłum.).

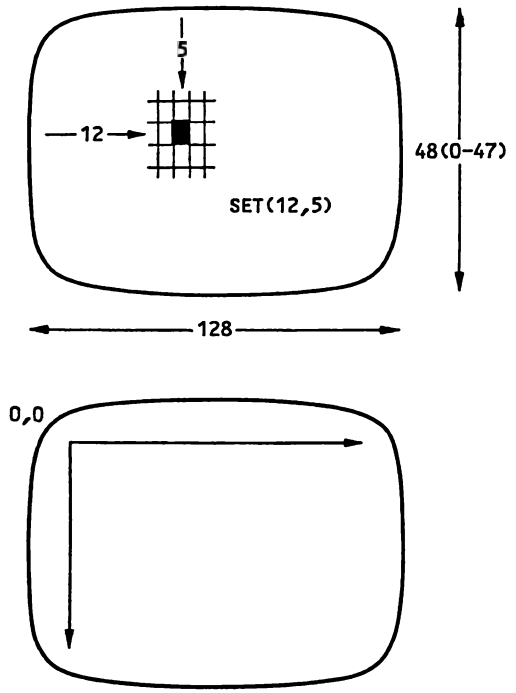
... w komputerze TRS-80

SET(X, Y): Zapalenie punktu o współrzędnych (X, Y)

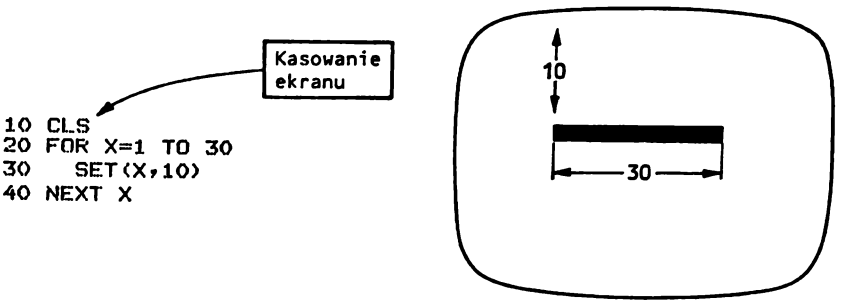
Ekran komputera TRS-80 jest podzielony na 128*48 punktów (128 na osi X, 48 na osi Y).
Dowolny z tych punktów może być zapalony **instrukcją SET(X,Y).**

Na przykład SET(12,5) zapala punkt w dwunastej kolumnie i w piątym wierszu.

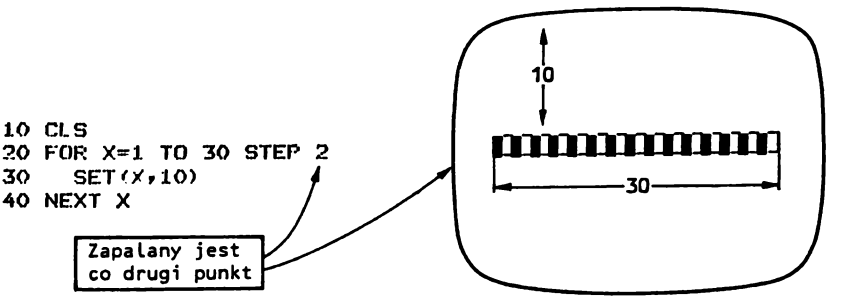
Uwaga: Wierzchołek układu współrzędnych (0,0) jest umieszczony na górze, po lewej stronie ekranu.



Kreślenie linii poziomej:



Kreślenie linii poziomej wykropkowanej:



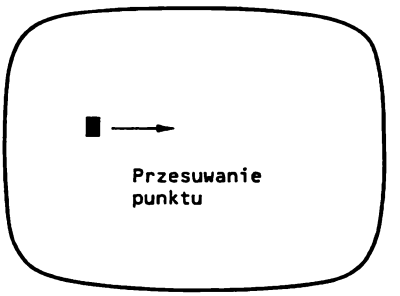
RESET(X,Y): Zgaszenie punktu o współrzędnych (X,Y)

Poniższy program zapala punkt (10,5) na okres dwóch sekund, a następnie go gasi:

```
10 SET (10,5)                    ' Zapalenie punktu 10,5
20 FOR TM=1 TO 200:NEXT TM       ' Opoznienie
30 RESET(10,5)                   ' Zgaszenie punktu 10,5
```

Ten natomiast przesuwa punkt po ekranie:

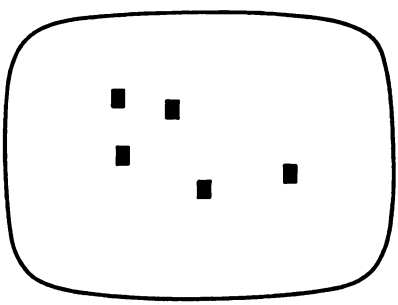
```
10 CLS
20 FOR X=1 TO 100
30  RESET(X-1,10)              ' Wymazanie poprzedniego punktu
40  SET(X,10)
50  FOR TM=1 TO 20:NEXT TM     ' Opoznienie
60 NEXT X
```



POINT(X,Y): Sprawdzenie, czy punkt jest zapalony

```
10 CLS
20 SET(10,20)
30 IF POINT(10,20)=-1 THEN PRINT "10-20 jest zapalony"
```

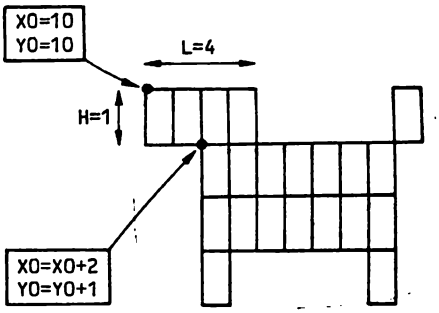
Poniższy program zapala przypadkowe punkty:



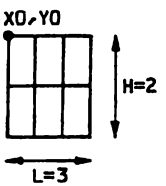
```
10 CLS
20 FOR G=1 TO 1000
30   X=RND(127)
40   Y=RND(47)
50   SET(X,Y)
60 NEXT G
70 GOTO 70       ' Czekanie (na nacisniecie BREAK)
```

Wykonywanie rysunku

Korzystając z podprogramu kreślenia wypełnionych prostokątów, narysujemy sylwetkę psa.

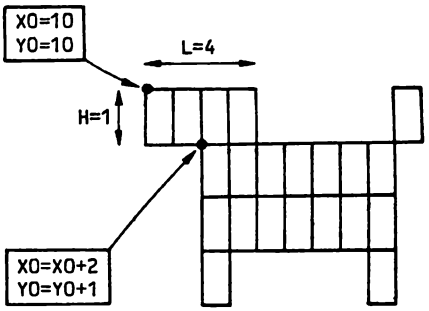



```
10 ' PIES
20 CLS
30 X0=10:Y0=10:L=4:H=1:GOSUB 120          ' Leb
40 X0=X0+2:Y0=Y0+1:L=7:H=2:GOSUB 120      ' Korpus
50 '
60 Y0=Y0+2:SET(X0,Y0)                     ' Lapy
70 X0=X0+6:SET(X0,Y0)
80 X0=X0+1:Y0=Y0-3:SET(X0,Y0)             ' Ogonek
90 '
100 GOTO 100
110 '
120 '----- Rysowanie pelnego prostokata
130 FOR Y1=Y0 TO Y0+H-1
140  FOR X1=X0 TO X0+L-1
150   SET(X1,Y1)
160  NEXT X1
170 NEXT Y1
180 RETURN
```

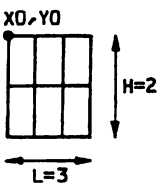


Wykonywanie rysunku

Korzystając z podprogramu kreślenia wypełnionych prostokątów, narysujemy sylwetkę psa.

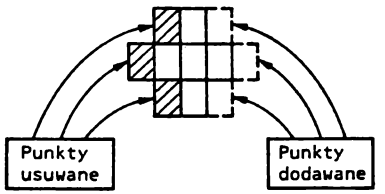


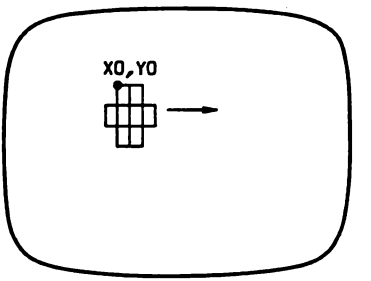

```
10 ' PIES
20 CLS
30 X0=10:Y0=10:L=4:H=1:GOSUB 120          ' Leb
40 X0=X0+2:Y0=Y0+1:L=7:H=2:GOSUB 120      ' Korpus
50 '
60 Y0=Y0+2:SET(X0,Y0)                     ' Lapy
70 X0=X0+6:SET(X0,Y0)
80 X0=X0+1:Y0=Y0-3:SET(X0,Y0)             ' Ogonek
90 '
100 GOTO 100
110 '
120 '----- Rysowanie pelnego prostokata
130 FOR Y1=Y0 TO Y0+H-1
140  FOR X1=X0 TO X0+L-1
150   SET(X1,Y1)
160  NEXT X1
170 NEXT Y1
180 RETURN
```



Animacja

Najprostszy sposób animacji figury na ekranie polega na kasowaniu starej i rysowaniu nowej figury.
Niestety, interpretery języka Basic nie działają dostatecznie szybko.
Zaleca się więc modyfikowanie figury przez usuwanie starych i dodawanie nowych punktów.






```
10 ' LATAJACY SPODEK
15 '
20 XO=10:YO=10:H=3:L=3
30 '---------- Rysunek poczatkowy
40 CLS
50 GOSUB 160
60 SET(XO-1,YO+1)
70 SET(XO+3,YO+1)
80 '---------- Animacja
90 XO=XO+1:IF XO>124 THEN STOP
100 RESET(XO-2,YO+1)                                    ' Usuwanie punktow
110 RESET(XO-1,YO)
120 RESET(XO-1,YO+2)
130 SET(XO+2,YO):SET(XO+2,YO+2):SET(XO+3,YO+1)  ' Dodawanie punktow
140 GOTO 90
150 '---------- Kreslenie pelnego prostokata
160 FOR Y1=YO TO YO+H-1
170  FOR X1=XO TO XO+L-1
180   SET(X1,Y1)
190  NEXT X1
200 NEXT Y
210 RETURN
```

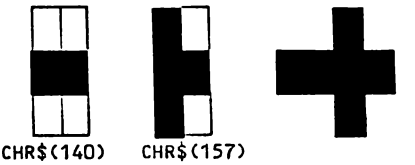
Adaptacja programu dla APPLE II:

```
40 GR:COLOR=3
60 PLOT XO-1,YO+1
70 PLOT XO+3,YO+1
90 XO=XO+1:IF XO>35 THEN STOP
100 COLOR=0:PLOT XO-2,YO+1
180 COLOR=3:PLOT X1,Y1
```

Zastosowanie znaków semigraficznych

Użycie znaków semigraficznych, składających się z sześciu punktów, zwiększa szybkość animacji, ponieważ jedną instrukcją PRINT możemy zapalić wiele punktów.

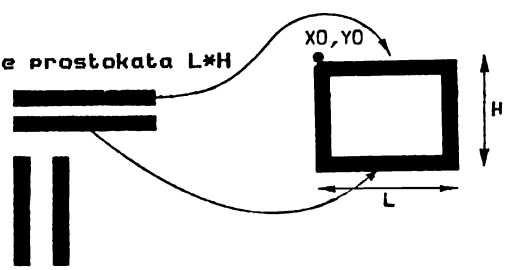
```
5 ' SAMOLOT
10 CLS
20 X$=" "+CHR$(140)+CHR$(157)
30 FOR I=1 TO 60
40  PRINT @I,X$;
50 NEXT I
60 GOTO 60
```



Kreślenie prostokątów

Poniższy program kreśli prostokąty.

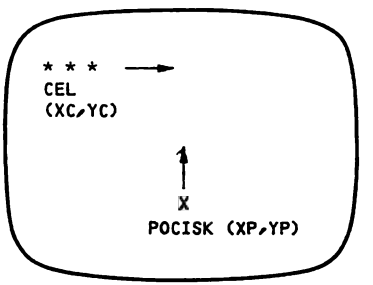
```
10 ' KOMORKA
20 CLS
30 X0=0:Y0=10:L=20:H=10:GOSUB 80
40 '----------
50 X0=30:Y0=30:GOSUB 80
60 GOTO 60
70 '---------- Rysowanie prostokata L*H
80 FOR X=X0 TO X0+L
90   SET (X,Y0)
100  SET(X,Y0+H)
110 NEXT X
120 FOR Y=Y0 TO Y0+H
130  SET(X0,Y)
140  SET(X0+L,Y)
150 NEXT Y
160 RETURN
```



Strzelanie do ruchomego celu

Cel przesuwa się po ekranie od strony lewej do prawej. Naciśnięcie dowolnego klawisza powoduje wystrzelenie pocisku przemieszczającego się od dołu ku górze ekranu.

Naciskając 'L' lub 'P' uzyskujemy odchylenie toru pocisku od pionu odpowiednio o−45 lub +45 stopni. Można wystrzelić wiele pocisków.

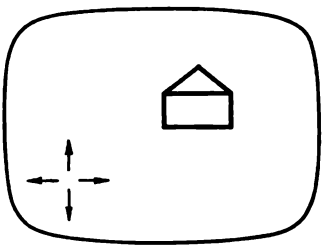



```
10 ' STRZELANIE DO CELU
20 CLEAR 1000
30 CLS
40 VC=1:XC=1:YC=8:PRINT @64*YC+XC-1,"***";
50 XP=30:YP=15:VP=-1:PRINT @64*YP+XP,"X";
60 '-----
70 X$=INKEY$:IF X$="" THEN 120          ' Oczekiwanie na strzal
80 ST=1:K=0                             ' Kat strzalu zero
90 IF X$="P" THEN K=-1
100 IF X$="L" THEN K=1
110 IF YP<YC THEN XP=30:YP=15:VP=-1:PRINT @64*YP+XP,"X";
120 '---------------- Przesuwanie celu
125 '
130 PRINT @64*YC+XC-1,"   ";           ' Kasowanie starego celu
140 XC=XC+VC
150 PRINT @64*YC+XC-1,"***";
160 '---------------- Ruch pocisku
170 IF ST=0 THEN 230
180 '--
190 PRINT @64*YP+XP," ";                ' Kasowanie starego pocisku
200 IF YP<YC THEN 230
210 YP=YP+VP:XP=XP+K:PRINT @64*YP+XP,"X";
220 '-------------------------------------
230 IF XC>XP+10 THEN PRINT @64*14,"PRZEGRANA";STOP
240 '---
250 IF YP=YC AND XP=>XC-1 AND XP<=XC+1 THEN
          PRINT @64*14,"TRAFIONY":STOP
260 '---
270 FOR TM=1 TO 10:NEXT TM              ' Opoznienie
280 GOTO 70
```

Rysownica

Zasada działania: Naciskając klawisze ↑↓←→ przesuwasz "punkt", który pozostawia za sobą ślad, co umożliwia wykonanie "odręcznego" rysunku.
Jeżeli naciśniesz klawisz 'P', to punkt będzie się przesuwać nie pozostawiając za sobą śladu, co umożliwia narysowanie figury składającej się z wielu rozłącznych części.

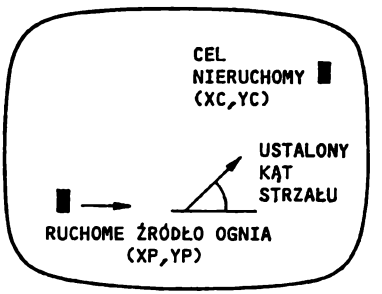
Ulepszenie: Rejestrowanie rysunku w tablicy, w celu uzyskania jego duplikatów w dowolnej skali.



```
10 REM       RYSOWNICA
20 :
25 CLS
30 X=10:Y=10
40 :
50 PRINT @64*13+30,"P:PODNIES PIORO";
60 PRINT @64*14+30,"O:OPUSC PIORO";
70 :
80 Z$=INKEY$:IF Z$<>"" THEN 130          ' Czekanie na nacisniecie klawisza
90 SET(X,Y)
100 RESET(X,Y)                           ' Migajacy kursor
110 GOTO 80
120 :
130 IF P=0 THEN SET(X,Y)
140 :
150 K=ASC(Z$)
160 IF K=8 THEN IF X>1 THEN X=X-1        ' W lewo
170 IF K=9 THEN IF X<125 THEN X=X+1
180 IF K=10 THEN IF Y<47 THEN Y=Y+1
190 IF K=91 THEN IF Y>1 THEN Y=Y-1
200 :
210 IF Z$="P" THEN P=1                   ' Podniesienie piora
220 IF Z$="O" THEN P=0
230 GOTO 80
```

Strzelanie do celu

A oto inny program strzelania do celu. Tym razem cel jest nieruchomy, natomiast źródło ognia (może to być na przykład czołg) porusza się.
Kąt strzału może być zmieniany przez modyfikację szybkości pocisku VP.

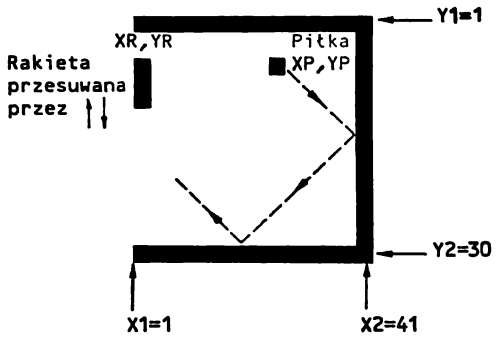



```
10 ' CZOLG       STRZELANIE DO CELU
20 CLS
30 XC=RND(10)+100:YC=5                  ' Wspolrzedne celu
40 SET(XC,YC)                           ' Wyswietlenie celu
50 VP=-1                                ' Szybkosc wlasna pocisku
60 XP=1:YP=20                           ' Wspolrzedne czolgu (pocisku)
70 '----
80 RESET(SX,SY)                         ' Kasowanie starej pozycji
90 SET(XP,YP)                           ' Nowa pozycja pocisku
100 SX=XP:SY=YP                         ' Poprzednia pozycja pocisku
110 IF YP<YC THEN PRINT @64*14+10,"PUDLO";:END
120 IF XP=XC AND YP=YC THEN PRINT @64*14+10,"TRAFIONY";:END
130 '----
140 X$=INKEY$:IF X<>"" THEN ST=1        ' Oddanie strzalu
150 IF ST=0 THEN 190
160 '----
170 YP=YP+VP                            ' Przesuniecie pocisku
180 '
190 XP=XP+1:GOTO 80
```

Adaptacja programu dla APPLE II: patrz "Grafika w komputerze APPLE II – wysoki poziom"

Squash

Piłka odbija się od trzech ścian.
Gracz odbija piłkę rakietą, którą przesuwa klawiszami ↑↓.



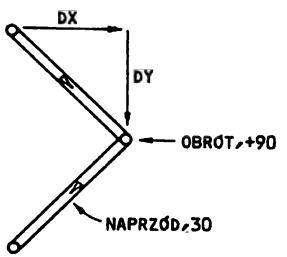

```
10 ' SQUASH  PILKA ODBIJANA OD SCIANY  ( TRS-80 )
20 CLS
30 X1=1:Y1=1:Y2=30
35 INPUT "POZIOM GRY (1,2,3): ";PG:X2=50-PG*3
40 '---------------------- Rysunek placu gry
50 FOR X=X1 TO X2:SET(X,Y1):SET(X,Y2):NEXT X
60 FOR Y=Y1 TO Y2:SET(X2,Y):NEXT Y
70 '---------------------- Rozpoczecie gry
75 ODB=0
80 DX=1:DY=1
110 XP=5:YP=20                ' Pilka
120 SET(XP,YP)
130 XR=2:YR=20                ' Rakieta
140 SET(XR,YR):SET(XR,YR+1):SET(XR,YR+2)
145 '
150 RESET(XP,YP)
160 '---------------------- Ruch pilki
170 XP=XP+DX:YP=YP+DY
180 SET(XP,YP)
190 IF XP=>X2-1 THEN DX=-DX
200 IF YP<Y1+2 THEN DY=-DY
210 IF YP=>Y2-1 THEN DY=-DY
220 '
230 IF XP<XR+2 AND YP=>YR AND YP<=YR+3 THEN DX=-DX:ODB=ODB+1
240 '
260 IF XP<XR+1 THEN PRINT @64*10+2,STR$(ODB)+"PUNKTOW";
    :FOR TM=1 TO 1000:NEXT TM:GOTO 20
270 '------------------ Przesuwanie rakiety
280 FOR XX=1 TO 5
290 X$=INKEY$:IF X="" THEN 330
310 IF ASC(X$)=10 THEN IF YR<Y2-3 THEN YR=YR+1:RESET(XR,YR-1)
                       :SET(XR,YR+2):GOTO 330
320 IF ASC(X$)=91 THEN IF YR>Y1 THEN YR=YR-1:RESET(XR,YR+3)
                       :SET(XR,YR):GOTO 330
330 NEXT XX
340 GOTO 150
```

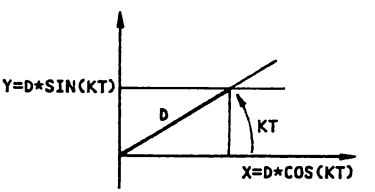
Adaptacja programu dla APPLE II: patrz "Grafika w komputerze APPLE II – wysoki poziom".

Kreślenie figur

Rozkazy graficzne występujące w większości języków programowania wymagają podawania współrzędnych (X,Y) kreślonych linii. Język LOGO charakteryzuje się oryginalnymi rozkazami graficznymi. Dwa z nich zasymulowaliśmy w języku BASIC:

NAPRZÓD odległość: Wykreślenie linii o długości równej 'odległości'.
OBRÓT kąt: Zmiana kierunku kreślenia o 'kąt'.






```
10 'SYMULATOR POLECEN LOGO
20 CLS
30 XO=30:YO=18
40 '----- Polecenia zdefiniowane w DATA (mozna zastapic przez INPUT)
50 DATA "OBROT",45
60 DATA "NAPRZOD",30
70 DATA "OBROT",+90
80 DATA "NAPRZOD",30
90 DATA "OBROT",+90
100 DATA "NAPRZOD",30
110 DATA "OBROT",+90
120 DATA "NAPRZOD",30
130 DATA "KONIEC"
140 '---------------- Interpretacja polecen
150 READ PL$
160 IF PL$="KONIEC" THEN 160
170 IF PL$="OBROT" THEN READ R:KT=KT+R:GOSUB 210
180 IF PL$="NAPRZOD" THEN READ ODL:GOSUB 290
190 GOTO 150
200 '---------------- Obliczenie nachylenia prostej
210 IF KT=>360 THEN KT=KT-360
220 KR=KT/360*6.28318
230 CX=COS(KR):CY=SIN(KR)
240 '
250 IF CX>-.01 AND CX<.01 THEN CX=0
260 IF CY>-.01 AND CY<.01 THEN CY=0
270 RETURN
```

Rysunek kwadratu

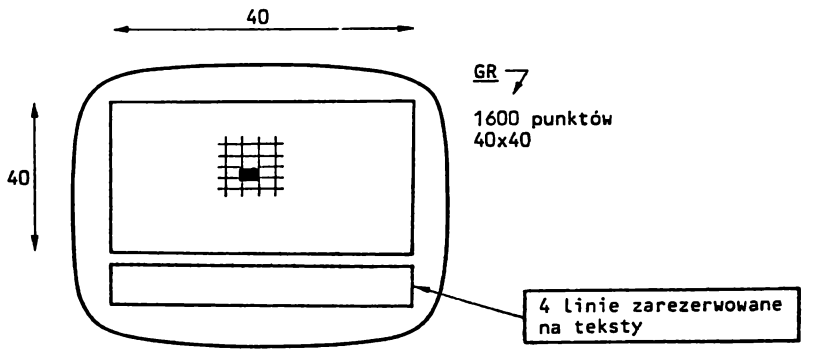
```
280 '----------------- Kreslenie prostej
290 '
300 DX=ODL*CX:DY=ODL*CY
310 FOR D=0 TO ODL
320  X=D*CX:Y=D*CY/2
330  SET(XO+X,YO+Y)
340 NEXT D
350 XO=XO+DX:YO=YO+DY/2
360 RETURN
370 ' Uwaga: Ponieważ jednostka na osi X jest dwukrotnie mniejsza
380 '        niż jednostka na osi Y, to w celu uzyskania
390 '        właściwych figur należy korygować wartość współrzędnej Y
400          (dzielenie przez 2).
```

Adaptacja programu dla APPLE II: patrz "Grafika w komputerze APPLE II — wysoki poziom".

... w komputerze APPLE II – niski poziom

GR: przejście do trybu graficznego niskiego poziomu

Instrukcja GR powoduje przejście do trybu graficznego niskiego poziomu. Ekran jest dzielony na $40 \times 40 = 1600$ punktów. 4 linie u dołu ekranu pozostają zarezerwowane dla tekstów.



TEXT: wyjście z trybu graficznego

Po instrukcji TEXT następuje powrót do trybu normalnego.

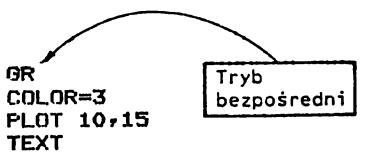
COLOR = nr koloru: wybranie koloru punktu

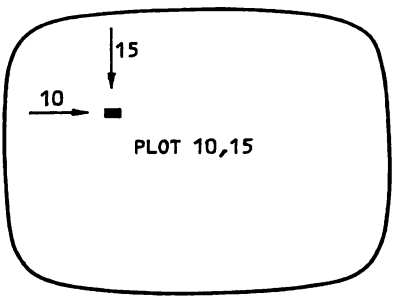
W grafice niskiego poziomu można używać szesnastu kolorów:

0 czarny
1 śliwkowy (magenta)
2 granatowy
3 fioletowy
4 ciemnozielony
5 szary
6 błękitny
7 jasnoniebieski
8 brązowy
9 pomarańczowy
10 szary
11 różowy
12 jasnozielony
13 żółty
14 turkusowy
15 biały

PLOT X,Y: zapalenie punktu

Plot X,Y zapala punkt w kolorze określonym instrukcją COLOR = nr koloru.





Wyświetlenie wszystkich kolorów:

Wyświetlanych jest 16 kolorów dostępnych w grafice niskiego poziomu.

```
10 GR
20 FOR K=0 TO 15
30    COLOR=K
40    PLOT 1,K
50 NEXT K
60 INPUT "Nacisnij <RETURN> ";X$   :REM Oczekiwanie na przejscie
70 TEXT                            :REM do trybu tekstowego
```

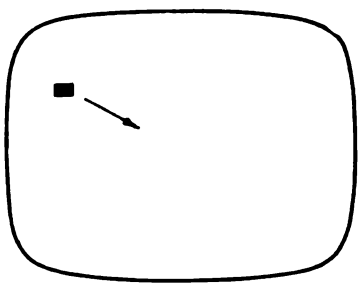
COLOR = 0; PLOT X,Y: zgaszenie punktu

Nie ma odrębnej instrukcji gaszącej punkt. W celu zgaszenia punktu, należy użyć ponownie instrukcji PLOT X,Y wybierając uprzednio kolor tła (czarny).

Przesuwanie punktu

Efekt przesunięcia uzyskamy gasząc zapalony uprzednio punkt i zapalając punkt sąsiedni.

```
10 GR                              :REM przejscie do trybu graficznego
30 FOR P=1 TO 30
40  COLOR=3:PLOT P,P               :REM zapalenie punktu P,P
50  FOR TM=1 TO 40:NEXT TM         :REM opoznienie
60  COLOR=0:PLOT P,P               :REM zgaszenie punktu P,P
70 NEXT P
80 INPUT "Nacisnij (RETURN)";X$
90 TEXT
```



SCRN (X,Y): badanie koloru punktu

Instrukcja ta umożliwia w szczególności stwierdzenie, czy określony punkt jest zapalony, czy też zgaszony (w kolorze tła).

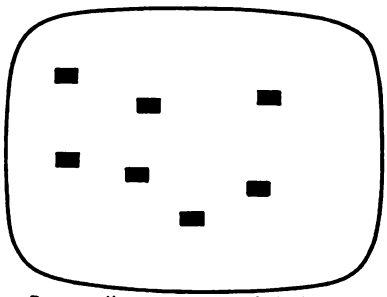
```
10 GR
20 COLOR=2
30 PLOT 10,15
40 PRINT SCRN(10,15)
50 GET A$
60 TEXT

RUN

2
```

Otrzymujemy numer wybranego koloru

Kolorowe gwiazdy na niebie

Zapalane są przypadkowe punkty ekranu. Kolory tych punktów są wybierane także w sposób losowy.

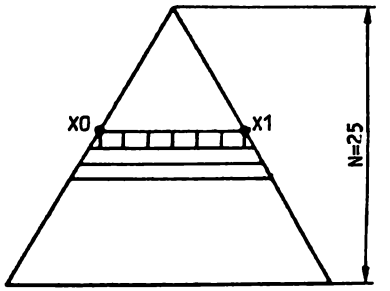
```
10 GR                        :REM przejscie do trybu graficznego
20 FOR G=1 TO 1000
30  COLOR=RND(1)*15+1        :REM przypadkowy kolor 1-15
40  X=RND(1)*40              :REM przypadkowa wartosc X  0 - 39
50  Y=RND(1)*40
60  PLOT X,Y
70 NEXT G
80 INPUT "Nacisnij <RETURN> ";X$
90 TEXT                      :REM powrot do trybu tekstowego
```



Przypadkowe punkty i kolory

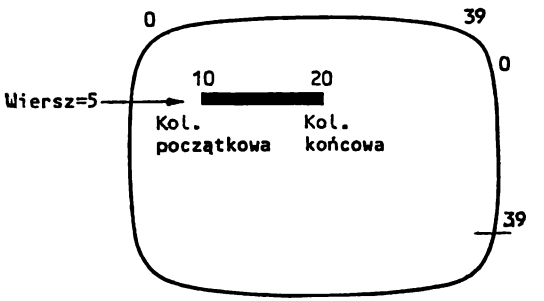
Wielobarwna piramida

Pętla wewnętrzna kreśli wielobarwną linię poziomą. Pętla zewnętrzna powtarza ten rysunek 25 razy, zwiększając stopniowo długość linii.
Wywołując podprogram z różnymi wartościami X0, Y0 otrzymamy zbiór wielobarwnych piramid.



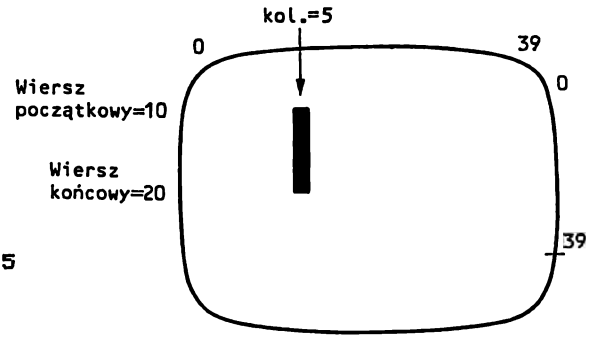

```
10 GR
20 X0=15:X1=15:Y0=0
30 N=25     :REM liczba linii
40 GOSUB 100
50 INPUT "Nacisnij <RETURN> ";X$
60 TEXT
70 END
90 REM ----------------------------------------
100 FOR Y=Y0 TO Y0+N          :REM N linii
110   X0=X0-.5:X1=X1+.5
120   FOR X=X0 TO X1          :REM rysowanie linii
130     COLOR=1+RND(15)
140     PLOT X,Y
150   NEXT X
160 NEXT Y
170 RETURN
```

HLIN kol. początkowa, kol. końcowa AT wiersz kreślenie linii poziomej




```
10 GR:COLOR=2

20 HLIN 10,20 AT 5

30 GET X$
40 TEXT
```

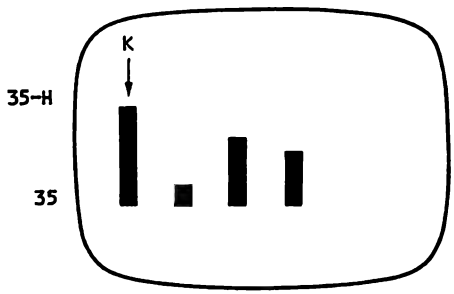
VLIN wiersz początkowy, wiersz końcowy AT kolumna:
kreślenie linii pionowej




```
10 GR:COLOR=2

20 VLIN 10,20 AT 5

30 GET X$
40 TEXT
```

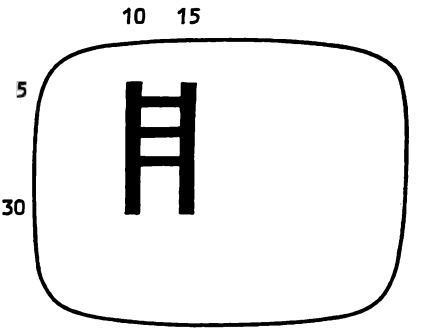
Histogram

Program wykreśla histogram dla czterech wartości zdefiniowanych w DATA.




```
10 DATA 15,3,10,8
20 :
30 GR:COLOR=3
40 FOR N=1 TO 4
50  READ H
60  K=N*4
70  VLIN (35-H),35 AT K
80 NEXT N
90 INPUT "Nacisnij <RETURN> ";X$
100 TEXT
```

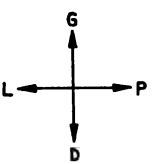
Rysunek drabiny

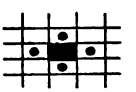



```
10 GR:COLOR=2
20 VLIN 5,30 AT 10
30 VLIN 5,30 AT 15
35 :
40 FOR S=7 TO 28 STEP 4
50  HLIN 10,15 AT S
60 NEXT S
70 INPUT "Nacisnij <RETURN> ";X$
80 TEXT
```

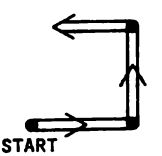
Rysunek zarejestrowany w DATA w formie przyrostowej

Rysunek jest zdefiniowany przez podanie kolejnych ruchów elementarnych. Określone są cztery kierunki ruchu. Nie możemy przesuwać punktu po przekątnych.

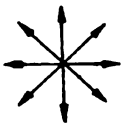


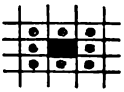


```
10 DATA P,P,P,G,G,G,L,L,L,D,D,D,"KONIEC"
20 :
30 GR:COLOR=3
40 READ KIER$:IF KIER$="KONIEC" THEN 150
50 :
60 IF KIER$="P" THEN X=X+1
70 IF KIER$="L" THEN X=X-1
80 IF KIER$="G" THEN Y=Y-1
90 IF KIER$="D" THEN Y=Y+1
100 :
110 PLOT X,Y
120 GOTO 40
130 :
150 INPUT "Nacisnij <RETURN> ";X$
160 TEXT
```



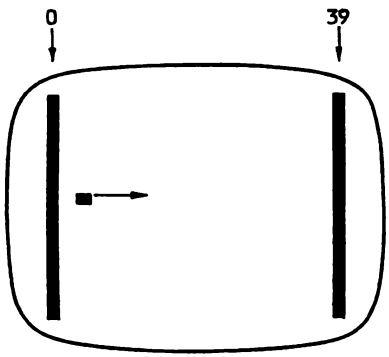
Moglibyśmy oczywiście zdefiniować 8 kierunków ruchu.





Odbijająca się piłka

Piłka odbija się od dwóch przeciwległych ścian.



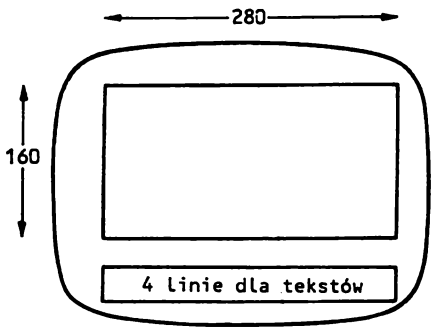

```
10 GR:COLOR=3
20 VLIN 0,39 AT 0              :REM rysowanie lewej sciany
30 VLIN 0,39 AT 39             :REM rysowanie prawej sciany
40 :
50 X=10:Y=20                   :REM wspolrzedne pilki
60 SZYBKOSC=-1                 :REM szybkosc pilki
70 :
80 X=X+SZYBKOSC
90 IF X<1 OR Y>38 THEN 200     :REM odbicie ?
100 :
110 COLOR=3:PLOT X,Y
120 FOR TM=1 TO 20:NEXT TM     :REM opoznienie
130 COLOR=0:PLOT X,Y
140 GOTO 80
150 :
200 SZYBKOSC=-SZYBKOSC          :REM odbicie pilki
210 GOTO 80
```

Aby przerwać program, naciśnij <Ctrl C>, a następnie napisz TEXT.

... w komputerze APPLE II – wysoki poziom

HGR: przejście do trybu graficznego wysokiego poziomu

Instrukcja HGR powoduje przejście do trybu graficznego wysokiego poziomu. Ekran jest dzielony na 280 × 160 punktów, które mogą być zapalane przez HPLOT X,Y. HCOLOR = kolor umożliwia wybranie koloru punktu.



HCOLOR = kolor: wybranie koloru

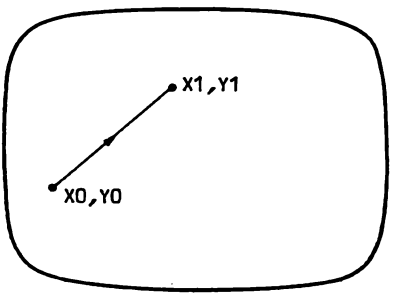
0 czarny
1 fioletowy
2 zielony
3 biały
4 czarny
5 niebieski
6 brązowy
7 biały

HPLOT X,Y: zapalenie punktu

Zapalany jest punkt w kolorze określonym uprzednio przez HCOLOR = kolor.

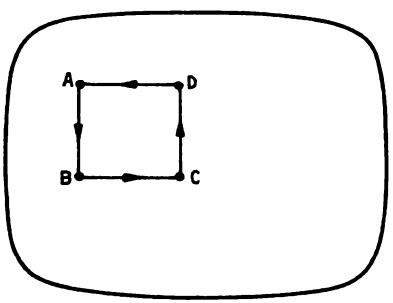
HPLOT x0,y0 TO x1,y1: kreślenie linii[1]

```
10 HGR
20 HCOLOR=3
30 HPLOT 10,100 TO 50,50
40 GET X$
50 TEXT
```



HPLOT x0,y0 TO x1,y1 TO x2,y2 ...: kreślenie linii łamanej

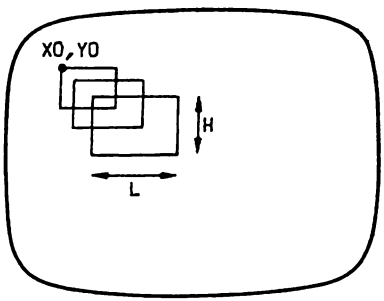
Linia łamana jest kreślona na podstawie podanych w instrukcji współrzędnych kolejnych punktów załamania.




```
10 HGR
20 HCOLOR=2
30 HPLOT 10,10 TO 10,50 TO 50,50 TO 50,10 TO 10,10
40 GET X$
50 TEXT
```

[1] Jeżeli w instrukcji kreślenia linii nie zostaną podane współrzędne x0, y0 punktu początkowego, to linia jest kreślona od ostatnio zapalonego punktu (przyp. tłum.).

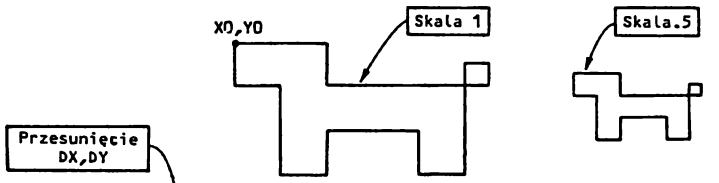
Poniższy program rysuje szereg prostokątów.




```
10 HGR
20 HCOLOR=3
30 H=10:L=15
40 XO=10:YO=10
45 MN=1.2
50 :
60 FOR N=1 TO 12
70  HPLOT XO,YO TO XO,YO+H TO XO+L,YO+H TO XO+L,YO TO XO,YO
75  XO=XO*MN:YO=YO*MN:L=L*MN:H=H*MN
80 NEXT N
100 :
110 INPUT "Nacisnij <RETURN> ";X$
120 TEXT
```

Rysunek w skali

Kreślona figura jest zdefiniowana w danych DATA przez określenie kolejnych ruchów. Dobierając skalę można uzyskać kopie rysunku w różnych rozmiarach.

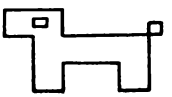



```
10 REM      RYSUNEK PRZYROSTOWY W SKALI
11 :
12 DATA 0,10,  10,0,  0,20,  10,0
20 DATA 0,-10, 20,0,  0,10
30 DATA 10,0,  0,-20, 3,0,   0,-3
40 DATA -3,0,  0,3,  -30,0,  0,-10
50 DATA -20,0
59 DATA 99,99
60 :
70 HGR:HCOLOR=3
80 XO=50:YO=50:SKL=1
90 GOSUB 200                  :REM 1-szy rysunek
100 :
110 RESTORE
120 XO=120:YO=70:SKL=.5
130 GOSUB 200                 :REM 2-gi rysunek
140 INPUT "Nacisnij <RETURN> ";X$
150 TEXT
160 END
170 REM ---------------- podprogram - rysunek
200 FOR P=1 TO 1000
210   READ DX:READ DY:IF DX=99 AND DY=99 THEN 270
220   X1=XO+DX*SKL
230   Y1=YO+DY*SKL
240   HPLOT XO,YO TO X1,Y1
250   XO=X1:YO=Y1
260 NEXT P
270 RETURN
```

Program kreśli figury ciągłe.

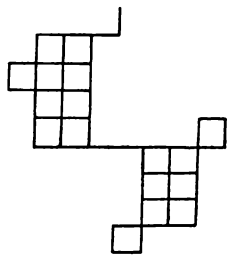
Uzupełnienie rysunku psa o oko możemy uzyskać sygnalizując występowanie nieciągłości przez wprowadzenie do danych DATA pary 0,0, i dołączając do programu linię 215:

```
 55 DATA 0,0,  10,3,  3,0
 56 DATA 0,3,  -3,0,  0,-3

215 IF DX=0 AND DY=0 THEN READ DX: READ DY
      :XO=XO+DX*SKL: YO=YO+DY*SKL: GOTO 260
```



Rysunek losowy

Kreślona figura jest kompozycją złożoną z odcinków o długości 5.



```
10 REM      RYSUNEK LOSOWY
20 :
30 HGR:HCOLOR=3
40 X0=120:Y0=80
50 X1=X0:Y1=Y0
60 :
70 FOR P=1 TO 1000
80 D=1+INT(RND(1)*4)
90 IF D=1 THEN X1=X0+5
100 IF D=2 THEN X1=X0-5
110 IF D=3 THEN Y1=Y0+5
120 IF D=4 THEN Y1=Y0-5
130 IF X1<0 OR X1>279 OR Y1<0 OR Y1>159 THEN 160
140 HPLOT X0,Y0 TO X1,Y1
150 X0=X1:Y0=Y1
160 NEXT P
```

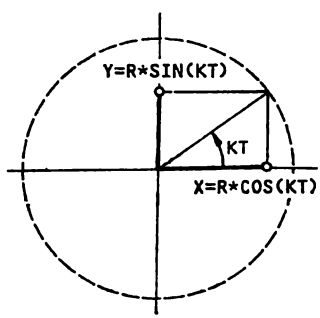
Aby uzyskiwać za każdym razem taki sam rysunek, należy inicjować ciąg liczb losowych instrukcję: 45 X = RND(N), gdzie N jest określoną liczbą ujemną.

Kreślenie okręgów i elips

Nie ma specjalnej instrukcji kreślącej okrąg. Należy kreślić go punkt po punkcie lub krótkimi odcinkami.
Kreślenie okręgu tymi metodami jest oczywiście o wiele wolniejsze niż gdyby było ono wykonywane bezpośrednio jedną instrukcją języka Basic (CIRCLE).

Współrzędne X,Y okręgu o promieniu R wyznaczamy na podstawie wzorów:

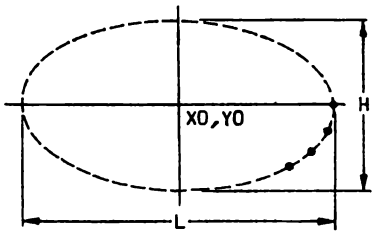
```
X = R * COS(KT)
Y = R * SIN(KT)
```



zmieniając KT od 0 do 6.28 radianów.

Dla elipsy, współprzędne X i Y są określone wzorem:

```
X = (L/2) * COS(KT)
Y = (H/2) * SIN(KT)
```



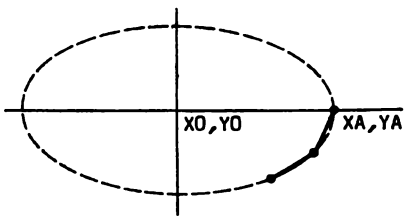

```
10 HGR: HCOLOR=3
20 L=100:H=50:XO=100:YO=100
30 GOSUB 100
40 GET X$
50 TEXT
60 END
90 REM ---------- kreslenie elipsy
100 FOR KT=0 TO 6.29 STEP .1
110   X=XO+(L/2)*COS(KT)
120   Y=YO+(H/2)*SIN(KT)
130   HPLOT X,Y
140 NEXT KT
150 RETURN
```

Dla L = H otrzymamy oczywiście okrąg o promieniu L/2[1].

Kreślenie elipsy odcinkami

Stosując tę metodę uzyskujemy większą szybkość kreślenia kosztem dokładności rysunku.

```
100 XA=XO+L/2:YA=YO
105 FOR KT=0 TO 6.29 STEP .3
110   X=XO+(L/2)*COS(KT)
120   Y=YO+(H/2)*SIN(KT)
130   HPLOT XA,YA TO X,Y
135   XA=X:YA=Y
140 NEXT KT
150 RETURN
```



Kreślenie figur geometrycznych wpisanych w okrąg

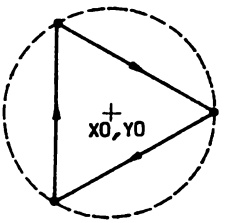
Przedstawiamy programy kreślące figury wpisane w okrąg lub w elipsę (trójkąt, kwadrat, romb).

---

[1] Ponieważ jednostka na osi X nie jest dokładnie równa jednostce na osi Y (tzn. 'punkt' na ekranie nie jest kwadratem lecz prostokątem), należy przeprowadzić korektę współrzędnej Y mnożąc ją przez 6/7. W programach przedstawionych w tym rozdziale przyjęto założenie, że 'punkt' ekranu jest kwadratem (przyp. tłum.).

Trójkąt

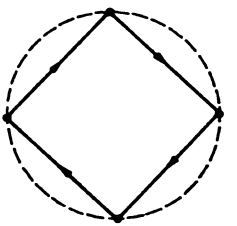
```
 10 HGR:HCOLOR=3
 20 XO=100:YO=100
 30 H=100:L=100
 40 LBOK=3            :REM liczba bokow
 50 GOSUB 100
 55 END
 60 :
 90 REM ---------------------
100 XA=XO+L/2:YA=YO
110 :
120 FOR B=1 TO LBOK
125  KT=B*6.28/LBOK
130  X=XO+L/2*COS(KT)
140  Y=YO+H/2*SIN(KT)
150  HPLOT XA,YA TO X,Y
155  XA=X:YA=Y
160 NEXT B
170 RETURN
```



Kwadrat

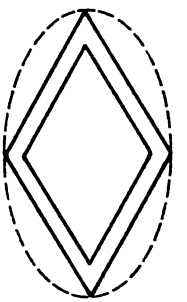
Wystarczy zmienić linię ' 40 LBOK = 4'.

```
40 LBOK=4
```



Romb

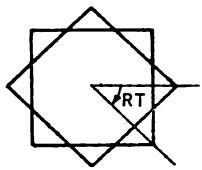
```
30 L=50:H=80
40 LBOK=4
50 GOSUB 100
51:
52 L=L*.8:H=H*.8
54 GOSUB 100
```



Obrót kwadratu

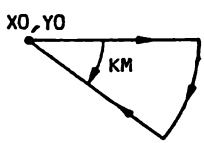
Program rysuje kwadrat obracający się wokół swego środka.

```
10 HGR:HCOLOR=3
20 X0=100:Y0=100
30 H=100:L=100
40 LBOK=4
50 FOR RT=0 TO 3.14 STEP 3.14/10
60  GOSUB 100
70 NEXT RT
80 GET X$
85 TEXT:END
90 REM ----------------------------
100 XA=X0+L/2*COS(RT):YA=Y0+H/2*SIN(RT)
110 :
120 FOR B=1 TO LBOK
125  KT=B*6.28/LBOK
130  X=X0+L/2*COS(KT)
140  Y=Y0+H/2*SIN(KT)
150  HPLOT XA,YA TO X,Y
155  XA=X:YA=Y
160 NEXT B
170 RETURN
```



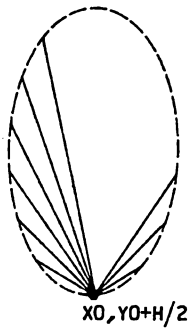
Kreślenie wycinka koła

```
10 HGR:HCOLOR=3
20 X0=100:Y0=100
30 R=50:KM=3.14/4
40 GOSUB 100
50 GET X$
60 TEXT
70 END
90 REM ------------------------
100 HPLOT X0,Y0 TO X0+R,Y0
110 :
120 FOR KT=0 TO KM STEP .05
130  X=X0+R*COS(KT)
140  Y=Y0+R*SIN(KT)
150  HPLOT X,Y
160 NEXT KT
170 HPLOT X,Y TO X0,Y0
180 RETURN
```



Inne rysunki

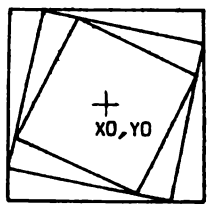
Jajko



```
 10 HGR:HCOLOR=3
 20 :
 30 X0=100:Y0=80
 40 L=100:H=150
 50 :
 60 FOR K=0 TO 3.14*2 STEP 3.14/50
 70   X=X0+L/2*COS(K)
 80   Y=Y0+H/2*SIN(K)
 90   HPLOT X0,Y0+H/2 TO X,Y
100 NEXT K
110 :
120 INPUT "Nacisnij <RETURN> ";X$
130 TEXT
```

Zagnieżdżone kwadraty

```
 10 HGR:HCOLOR=3
 20 M=.1
 30 X0=100:Y0=100
 40 X1=60:Y1=60
 50 :
 60 FOR K=1 TO 20
 70  HPLOT X0+X1,Y0-Y1
 80  HPLOT TO X0+Y1,Y0+X1
 90  HPLOT TO X0-X1,Y0+Y1
100  HPLOT TO X0-Y1,Y0-X1
110  HPLOT TO X0+X1,Y0-Y1
120  :
130  X1=X1-(X1-Y1)*M
140  Y1=Y1-(X1+Y1)*M
150 NEXT K
```



Kardioida

```
10 HGR:HCOLOR=3
20 X0=100:Y0=70
30 FOR K=0 TO 6.29 STEP 3.14/50
40    X=50*(1-COS(K))*COS(K)
50    Y=50*(1-COS(K))*SIN(K)
60    HPLOT X0+X,Y0+Y
70 NEXT K
```

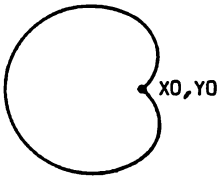
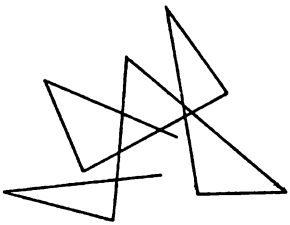


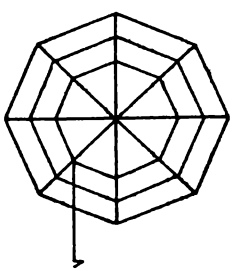
Figura losowa

```
10 HGR:HCOLOR=3
20 HPLOT 50,50
30 FOR P=1 TO 50
40  X=RND(1)*150
50  Y=RND(1)*150
60  HPLOT TO X,Y
70 NEXT P
```



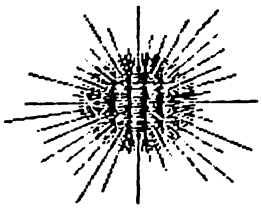
Pajęczyna

```
 10 HGR:HCOLOR=3
 20 X0=60:Y0=53
 30 R=11
 40 FOR M=4 TO 1 STEP -.5
 50   XA=X0+R*M:YA=Y0
 60   FOR K=0 TO 6.29 STEP 6.28/8
 70     X=X0+R*M*COS(K)
 80     Y=Y0+R*M*SIN(K)
 90     HPLOT XA,YA TO X,Y
100     IF M=4 THEN HPLOT X0,Y0 TO X,Y
110     XA=X:YA=Y
120   NEXT K
130 NEXT M
140 :
150 FOR Y=90 TO 150 STEP 3:HPLOT 50,Y:NEXT Y
160 HPLOT TO 52,Y:HPLOT TO 48,Y+1
```



Słońce

```
 10 HGR:HCOLOR=3
 20 X0=100:Y0=100
 30 :
 40 FOR K=0 TO 6.29 STEP 3.14/50
 50  X=30*COS(K)+X0
 60  Y=30*SIN(K)+Y0
 70  HPLOT X0,Y0 TO X,Y
 80 NEXT K
 90 :
100 FOR K=0 TO 6.29 STEP 3.14/20
110  R=30+40*RND(1)
120  X=R*COS(K):Y=R*SIN(K)
130  HPLOT X0,Y0 TO X0+X,Y0+Y
140 NEXT K
```



Squash (patrz str. 127, 128)

```
 10 REM        SQUASH     ( APPLE - niski poziom )
 20 :
 30 REM ---------  Rysunek placu gry
 35 INPUT "POZIOM GRY (1,2,3): ";P:X2=35-P*3
 40 X1=2:Y1=2:Y2=35
 50 HOME: GR: COLOR=3: VTAB 22: PRINT "(-- --) PORUSZANIE RAKIETY"
 60 HLIN X1,X2 AT Y1: HLIN X1,X2 AT Y2
 70 VLIN Y1,Y2 AT X2
 75 ODB=0
 90 DX=1:DY=1
110 XP=5:YP=20
120 PLOT XP,YP
130 XR=2:YR=20
140 VLIN YR,YR+2 AT XR
150 COLOR=0: PLOT XP,YP
160 REM ---------------------- Ruch pilki
170 XP=XP+DX:YP=YP+DY
180 COLOR=3: PLOT XP,YP
190 IF XP=>X2-1 THEN DX=-DX: REM Odbicie od przedniej sciany
200 IF YP<Y1+2 THEN DY=-DY:  REM Odbicie od scian bocznych
210 IF YP=>Y2-1 THEN DY=-DY
220 :
230 IF XP<XR+2 AND YP=>YR AND YP<=YR+3 THEN DX=-DX:ODB=ODB+1
240 REM ---------- Utrata pilki
260 IF XP<XR+1 THEN PRINT ODB;"PUNKTOW": FOR TM=1 TO 3000:NEXT TM: GOTO 40
270 REM --------------- Przesuwanie rakiety
280 FOR XX=1 TO 3
290 K=PEEK(-16384): IF K<128 THEN 330
300 POKE -16368,0
310 IF K-128=8 THEN IF YR<Y2-3 THEN YR=YR+1: COLOR=0: PLOT XR,YR-1:
                                     COLOR=3: PLOT XR,YR+2
320 IF K-128=21 THEN IF YR>Y1 THEN YR=YR-1: COLOR=0: PLOT XR,YR+3:
                                    COLOR=3: PLOT XR,YR
330 NEXT XX
340 GOTO 150
```

Kreślenie figur geometrycznych (patrz str. 129)

```
10 REM SYMULATOR POLECEN LOGO (APPLE - wysoki poziom)
20 HGR : HCOLOR= 3
25 CX = 1:CY = 0
30 X0 = 80:Y0 = 80
50 DATA  "NAPRZOD",30
60 DATA  "OBROT",90
70 DATA  "NAPRZOD",30
80 DATA  "OBROT",90
90 DATA  "NAPRZOD",30
100 DATA "OBROT",90
110 DATA "NAPRZOD",30
130 DATA "KONIEC"
140 READ PL$
160 IF PL$ = "KONIEC" THEN 350
170 IF PL$ = "OBROT" THEN READ R:KT = KT + R: GOSUB 210
180 IF PL$ = "NAPRZOD" THEN READ ODL: GOSUB 300
185 PRINT PL$
190 GOTO 140
200 REM --------- Obliczanie nachylenia prostej
210 IF KT => 360 THEN KT = KT - 360
220 KR = KT / 360 * 6.28318
230 CX = COS(KR):CY = SIN(KR)
240 IF CX > -.01 AND CX < .01 THEN CX = 0
270 RETURN
290 REM ---------- Kreslenie prostej
300 DX = ODL * CX:DY = ODL * CY
310 HPLOT X0,Y0 TO X0 + DX,Y0 + DY
320 X0 = X0 + DX:Y0 = Y0 + DY
330 RETURN
350 GET X$
360 TEXT
```

Rysownica (patrz str. 126)

```
10 REM      RYSOWNICA  ( APPLE - niski poziom )
20 :
30 GR: HOME: VTAB 21
40 PRINT "RYSUJ PRZY POMOCY CZTERECH STRZALEK "
50 PRINT " ( APPLE2+: (-- --) ( ) )"
60 PRINT "P:PODNIES PIORO / O:OPUSC PIORO "
70 REM ----------------------------------------
80 X=10:Y=10: COLOR=3
100 REM ------------------- Migajacy kursor
110 K=PEEK(-16384): IF K=>128 THEN 150
120 COLOR=3: PLOT X,Y: COLOR=0: PLOT X,Y
130 GOTO 100
140 REM -------------------------
150 IF P=0 THEN COLOR=3: PLOT X,Y
160 POKE -16368,0
170 :
180 K=K-128:Z$=CHR$(K)
190 IF K=8 THEN IF X>1 THEN X=X-1
200 IF K=21 THEN IF X<38 THEN X=X+1
210 IF K=10 OR K=44 THEN IF Y<38 THEN Y=Y+1
220 IF K=11 OR K=46 THEN IF Y>1 THEN Y=Y-1
240 IF Z$="P" THEN P=1
250 IF Z$="O" THEN P=0
260 GOTO 100
```

Kreślenie krzywej (patrz str. 122)

```
10 REM KRESLENIE KRZYWEJ (APPLE - wysoki poziom)
20 :
30 HEKR=150:LEKR=200
40 INPUT "Granica 1? ";G1
50 INPUT "Granica 2? ";G2
60 INPUT "Krok? ";KROK
70 :
75 X=G1: GOSUB 290:Y1=Y:Y2=Y
80 FOR X=G1 TO G2 STEP KROK
90 : GOSUB 290
100 : IF Y<Y1 THEN Y1=Y
110 : IF Y>Y2 THEN Y2=Y
120 NEXT X
135 :
140 HGR: HCOLOR=3
150 SX=LEKR/(G2-G1): REM Skala X
160 SY=(HEKR-2)/(Y2-Y1): REM Skala Y
180 IF Y2=>0 AND Y1<=0 THEN Y=HEKR-ABS(Y1)*SY: HPLOT 1,Y TO LEKR,Y
190 IF G1<=0 AND G2=>0 THEN X=ABS(G1)*SX: HPLOT X,1 TO X,HEKR
200 REM ------
210 FOR X=G1 TO G2 STEP KROK
220 : GOSUB 290
230 : PX=(X-G1)*SX
240 : PY=HEKR-(Y-Y1)*SY
250 : HPLOT PX,PY
260 NEXT X
265 PRINT "Y1=";Y1;"Y2=";Y2
270 INPUT "Nacisnij (RETURN) ";X$
275 TEXT: END
280 REM ------------ Krzywa do wykreslenia
290 Y=SIN(X)*X
300 RETURN
```

Strzelanie do celu (patrz. str. 127)

```
10 REM CZOLG  STRZELANIE DO CELU (APPLE - niski poziom)
20 :
30 GR:WK=0
40 XC=INT(RND(1)*5)+20:YC=5
50 COLOR=3: PLOT XC,YC
55 VP=-1
60 XP=1:YP=20
70 :
80 COLOR=0: PLOT SX,SY
90 COLOR=3: PLOT XP,YP
100 SX=XP:SY=YP
110 IF YP<YC THEN PRINT "PUDLO": FOR TM=1 TO 500: NEXT TM: GOTO 30
115 IF XP>38 THEN PRINT "PUDLO": STOP
120 IF XP=XC AND YP=YC THEN PRINT "TRAFIONY": STOP
130 :
140 K=PEEK(-16384): IF K>128 THEN POKE -16364,0: WK=1
150 IF WK=0 THEN 190
160 :
170 YP=YP+VP
180 :
190 XP=XP+1: GOTO 80
```

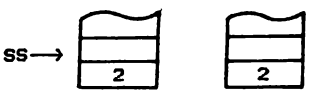
Mini-interpreter pętli

Chcąc objaśnić działanie interpretera języka Basic, zapisaliśmy w DATA krótki pseudo-program.

```
30 REM ---- interpretowany pseudo-program
40 DATA "POWTARZAJ 2"
50 DATA "Duza petla"
60 DATA "POWTARZAJ 3"
70 DATA "Petla wewnetrzna"
80 DATA "JESZCZERAZ"
90 DATA "JESZCZERAZ"
100 DATA *
110 REM ------------------------------------------
120 FOR I=1 TO 100
130  READ X$:IF X$="*" THEN LP=I-1:GOTO 170
140  PL$(I)=X$
150 NEXT I
160 :
170 WL=0        :REM wskaznik biezacej linii
180 :
190 LB$=PL$(WL)
200 P$=LEFT$(LB$,4)
210 IF P$="POWT" THEN GOTO 330
220 :
230 IF P$="JESZ" THEN GOTO 370
240 :
250 PRINT LB$
260 :
270 WL=WL+1
280 IF WL>LP THEN STOP
290 GOTO 190
300 REM ---------------------------------- Powtarzaj
310 REM  SS --> strzalka stosu PT() z numerami linii po POWTARZAJ
320 REM                        LC() z licznikami petli
330 FOR P=1 TO LEN(LB$)
332  IF MID$(LB$,P,1)=" " THEN 340
334 NEXT P
336 P=0
340 SS=SS+1:PT(SS)=WL+1:LC(SS)=VAL(RIGHT$(LB$,LEN(LB$)-P))
350 WL=WL+1:GOTO 190
360 REM ---------------------------------- Jeszcze raz
370 LC(SS)=LC(SS)-1:IF LC(SS)>0 THEN WL=PT(SS):GOTO 190
380 SS=SS-1:WL=WL+1:GOTO 190
```

```
Tablica PL$()
1  POWTARZAJ 2
2  Duza petla
3  POWTARZAJ 3
4  Petla wewnetrzna
5  JESZCZERAZ
6  JESZCZERAZ
```

WL⟶ 4



Stos PT() nr linii — Stos LC() licznik pętli

```
Duza petla
Petla wewnetrzna
Petla wewnetrzna
Petla wewnetrzna
Duza petla
Petla wewnetrzna
Petla wewnetrzna
Petla wewnetrzna
```

Mini-interpreter LOGO

Zinterpretujemy w języku Basic kilka rozkazów graficznych języka LOGO.

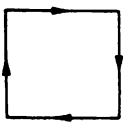
Podstawowe instrukcje

ODLEGŁOŚĆ 10	Ustalenie bieżącej odległości na 10
NAPRZÓD	Przesunięcie na odległość równą odległości bieżącej
NAPRZÓD 10	Przesunięcie na odległość 10
OBRÓT 90	Zmiana kierunku kreślenia o 90 stopni
ODL+3	Zwiększenie odległości bieżącej o 3
NIEMALUJ	Wyłączenie kreślenia
MALUJ	Ponowne włączenie kreślenia

Pętle

Instrukcje zawarte między 'POWTARZAJ X' a 'JESZCZERAZ' są wykonywane X razy.

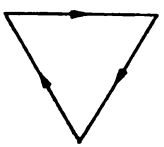
```
POWTARZAJ 4
 OBROT 90
 NAPRZOD
JESZCZERAZ
```



Procedury

"Procedury" deklaruje się w następujący sposób:

```
DEFI TROJKAT
POWTARZAJ 3
 NAPRZOD
 OBROT 120
JESZCZERAZ
KONIEC
```



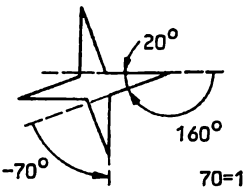
Aby uzyskać rysunek trójkąta, wystarczy następnie napisać:

```
TROJKAT
```

Podczas czytania instrukcji pseudo-programu wywołanie procedury jest zastępowane przez ciąg instrukcji podany w jej deklaracji.

Gwiazda czteroramienna

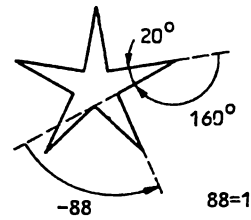
```
POWTARZAJ 4
 NAPRZOD 10
 OBROT 160
 NAPRZOD 10
 OBROT -70
JESZCZERAZ
```



70=180-(360/4)-20

Gwiazda pięcioramienna

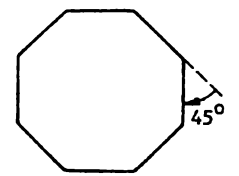
```
POWTARZAJ 5
 NAPRZOD 10
 OBROT 160
 NAPRZOD 10
 OBROT -88
JESZCZERAZ
```



88=180-(360/5)-20

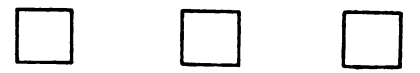
Ośmiokąt

```
POWTARZAJ 8
 NAPRZOD 10
 OBROT 45
JESZCZERAZ
```



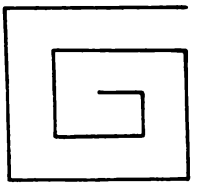
Trzy kwadraty

```
POWTARZAJ 3
 ODLEGLOSC 5
 MALUJ
 KWADRAT
 NIEMALUJ
 NAPRZOD 15
JESZCZERAZ
```



Spirala

```
ODLEGLOSC 5
POWTARZAJ 6
 NAPRZOD
 OBROT 90
 ODL+ 2
JESZCZERAZ
```



```
10 REM LOGO   SYMULATOR LOGO (APPLE)
20 :
50 DIM PL$(100)
60 X0=100:Y0=100:PX=1:PY=0:ODL=20
65 HGR:HCOLOR=3
70 REM ------- interpretowany pseudoprogram ---
80 :
90 DATA "DEFI KWADRAT"
100 DATA "POWTARZAJ 4"
110 DATA "OBROT 90"
120 DATA "NAPRZOD"
130 DATA "JESZCZERAZ"
140 DATA "KONIEC"
150 :
200 DATA "ODLEGLOSC 50"
210 DATA "KWADRAT"
220 DATA "NIEMALUJ"
230 DATA "NAPRZOD 60"
240 DATA "MALUJ"
250 DATA "ODLEGLOSC 30"
260 DATA "POWTARZAJ 8"
270 DATA "KWADRAT"
280 DATA "OBROT 45"
290 DATA "JESZCZERAZ"
510 DATA *
```

```
520 REM ------------ Czytanie polecen -------
530 I=1
540 :
550 READ X$:PRINT X$
560 :
570 IF LEFT$(X$,4)="DEFI" THEN GOSUB 1250:GOTO 550
580 :
590 IF X$="*" THEN LP=I:GOTO 670
600 FOR K=1 TO 5
610  IF X$=F$(K,1) THEN GOSUB 1330
620 NEXT K
630 :
640 PL$(I)=X$
650 I=I+1:GOTO 550
660 :
670 WL=1:REM wskaznik aktualnej linii
680 :
690 LB$=PL$(WL)
700 P$=LEFT$(LB$,4)
710  IF P$="POWT" THEN GOTO 930
720  IF P$="JESZ" THEN GOTO 970
730  IF P$="NAPR" THEN GOSUB 1080
740  IF P$="OBRO" THEN GOSUB 1000
750  IF P$="ODLE" THEN GOSUB 860:IF V<>0 THEN ODL=V
760  IF P$="ODL+" THEN GOSUB 860:ODL=ODL+V
770  IF P$="NIEM" THEN NM=1
780  IF P$="MALU" THEN NM=0
790 WL=WL+1:IF WL>LP THEN END
800 GOTO 690
850 REM -- Wydzielanie wartosci liczbowej z polecenia --
860 V=0
865 FOR P=1 TO LEN(LB$)
870 IF MID$(LB$,P,1)=" " THEN GOTO 895
880 NEXT P
890 P=0:RETURN
895 V=VAL(RIGHT$(LB$,LEN(LB$)-P)):RETURN
900 REM ---------------- Powtarzaj -------------------------
910 REM  SS- strzalka stosow:
915           PT() z numerami linii po POWTARZAJ
920           LC() z licznikami petli
930 GOSUB 860
940 SS=SS+1:PT(SS)=WL+1:LC(SS)=V
950 WL=WL+1:GOTO 690
960 REM ------------- Jeszcze raz ----------------------
970 LC(SS)=LC(SS)-1:IF LC(SS)>0 THEN WL=PT(SS):GOTO 690
980 SS=SS-1:WL=WL+1:GOTO 690
990 REM ------ Wyznaczanie kierunku kreslenia ----------
1000 GOSUB 860
1010 KT=KT+V
1020 IF KT=>360 THEN KT=KT-360
1030 KR=KT/360*6.28318
1040 PX=COS(KR):PY=SIN(KR)
1045 IF PX>-.01 AND PX<.01 THEN PX=0
1050 RETURN
```

```
1060 REM ------ Kreslenie prostej ----------------------
1070 :
1080 GOSUB 860
1090 IF P=0 THEN 1110
1100 ODL=V
1110 DX=ODL*PX:DY=ODL*PY
1115 IF NM=1 THEN 1180
1120 HPLOT X0,Y0 TO X0+DX,Y0+DY
1180 X0=X0+DX:Y0=Y0+DY
1190 RETURN
1240 REM ---- Gromadzenie definicji funkcji w F$(,) ------
1250 NF=NF+1
1260 F$(NF,1)=RIGHT$(X$,LEN(X$)-5)
1270 FOR P=2 TO 10
1280  READ F$(NF,P)
1290  IF F$(NF,P)="KONIEC" THEN RETURN
1300 NEXT P
1310 STOP
1320 REM ----------- Rozwijanie funkcji --------
1330 P=2
1340 :
1350 IF F$(K,P)="KONIEC" THEN RETURN
1360 PL$(I)=F$(K,P)
1370 P=P+1:I=I+1:GOTO 1350
```

Tablica 1

Równoważne operacje graficzne TRS-80/APPLE II

	TRS-80	APPLE NP[1]	APPLE WP[1]
Przejście do trybu graficznego		GR TEXT	HGR TEXT
Kasowanie ekranu	CLS	HOME	HOME
Wybieranie koloru		COLOR = kolor	HCOLOR = kolor
Zapalenie punktu	SET (X, Y)	PLOT X, Y	HPLOT X, Y
Zgaszenie punktu	RESET (X, Y)	COLOR = 0 PLOT X, Y	HCOLOR = 0 HPLOT X, Y
Kreślenie prostej	FOR I = 1 TO N X = Y = SET (X, Y) NEXT I	HLIN k1, k2 AT w VLIN w1, w2 AT k	HPLOT x0, y0 TO x1, y1

[1] NP – niski poziom
WP – wysoki poziom

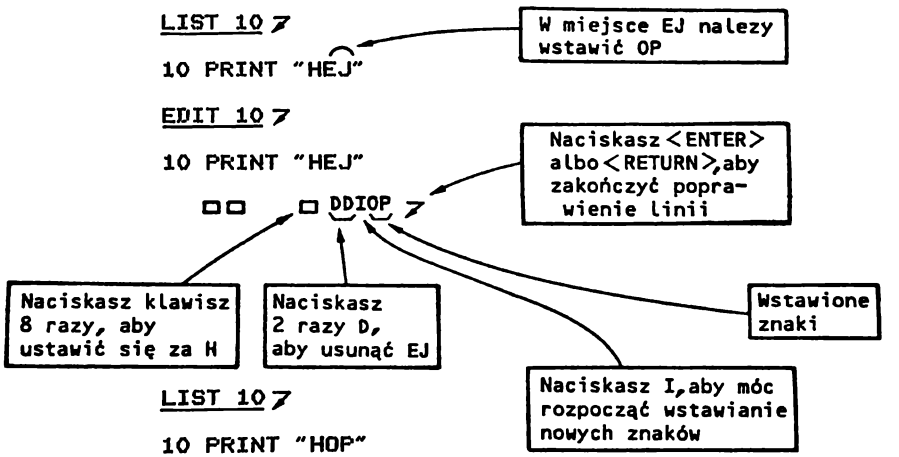
Dodatek I

Edytor

Edytor linii TRS-80 i Microsoft 5.

Modyfikacja programu polega na zmianie poszczególnych linii. Przedstawiamy prosty przypadek modyfikacji:

Zamienimy 10 PRINT "HEJ" na 10 PRINT "HOP"

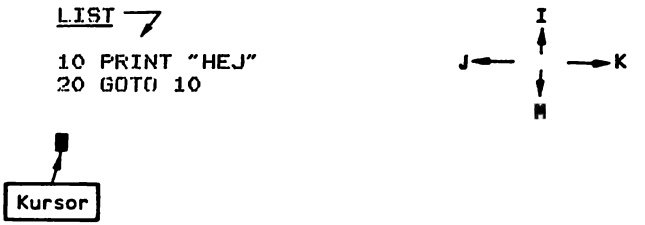


Modyfikacja linii programu w APPLE

Przed przystąpieniem do modyfikacji linii należy ustawić kursor na jej początku.

Ustawienie kursora

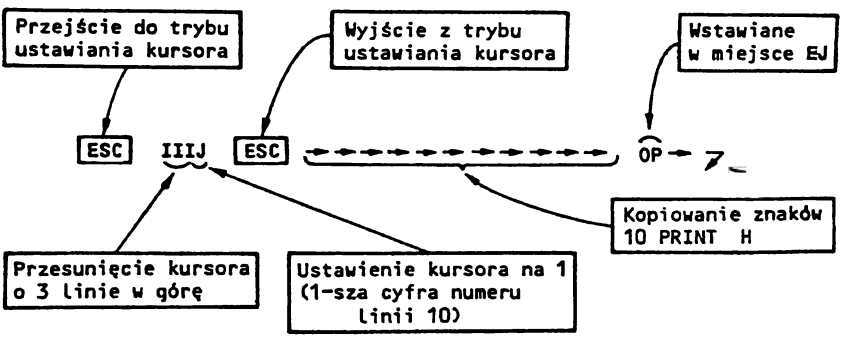
Kursor jest ustawiany za pomocą klawisza <ESCAPE> i czterech klawiszy J, I, K, M.



Modyfikacja

Po ustawieniu kursora na początku linii modyfikuje się ją w następujący sposób:

- Znaki niezmieniane "kopiuje" się za pomocą →
- Nowe znaki wprowadza się w miejsce starych.
- ← kasuje znak.

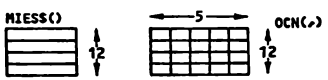



```
LIST

10 PRINT "HOP"
20 GOTO 10
```

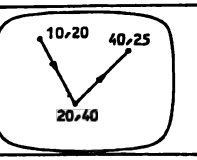
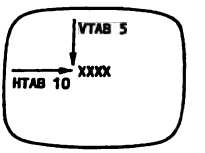
Dodatek II

Zestawienie instrukcji języka Basic

	ABS(−5)	Wyznacza absolutną wartość liczby PRINT ABS(−5) → 5
	AND	Umożliwia badanie wielu warunków: Instrukcje po THEN są wykonywane JEZELI warunek 1 I warunek 2 są spełnione: 90 INPUT "X?"; X 100 IF (X > 0) AND (X < 10) THEN PRINT "X jest większe niż 0 i mniejsze niż 10"
	ASC("B") ASC("CHARLIE")	Określa kod ASCII podanego znaku albo pierwszego znaku tekstu (tekst nie może być pusty) PRINT ASC("B") → 66 PRINT ASC("CHARLIE") → 67
	ATN(X)	Wyznacza arcus tangens X wyrażony w radianach
	AUTO 100 AUTO 100,5	Powoduje automatyczną numerację linii programu poczynając od 100 co 10 Powoduje automatyczną numerację od 100 co 5
	CHR$(66) CHR$(7)	Określa znak o kodzie równym 66 (tzn B) PRINT CHR$(66) wyświetla znak B PRINT CHR$(7); uruchamia sygnał dźwiękowy
	CLEAR 1000	– Zeruje zmienne – Rezerwuje 1000 znaków na teksty
TRS80	CLS	Kasuje ekran
APPLE	COLOR = kolor	Ustala kolor punktów (0 – 15), które będą wyświetlane przez PLOT X, Y
	CONT	Wznawia wykonywanie programu przerwanego przez <BREAK> lub <Ctrl C> albo przez umieszczoną w programie instrukcję STOP

	COS(X)	Wyznacza cosinus X wyrażony w radianach
	DATA "Styczen:", 85	Definiuje dane odczytywane instrukcją READ 10 DATA "Styczen:", 85 20 READ MIES$: READ ROK 30 PRINT MIES$, ROK → Styczen: 85
TRS80 M5.	DEF FNSR(A, B) = (A + B)/2	Definiuje funkcję jednej lub wielu zmiennych, która może być następnie wywoływana w programie 10 DEF FNSR(A, B) = (A + B)/2 20 PRINT FNSR(8, 10) → 9 Zaokrąglanie: 10 DEF FNZG = INT(X+.5) 20 PRINT FNZG(12.4) → 12 30 PRINT FNZG(12.6) → 13 10 DEF FNZG(X) = INT(X*100+.5)/100 20 PRINT FNZG(12.345) → 12.35
APPLE	DEF FNM(A) = A*2	Definiuje funkcję jednej zmiennej
APPLE	DEL 10 DEL 10,30	Usuwa linię 10 Usuwa linie o numerach od 10 do 30
M.5 TRS80	DELETE 10 DELETE 10-30 DELETE 100-	Usuwa linię 10 Usuwa linie o numerach od 10 do 30 Usuwa końcowe linie programu, od linii o numerze 100
	DIM MIES$(12) DIM OCN(12,5)	Rezerwuje obszar pamięci dla tablicy MIES$( )  (Tablicy, której rozmiar nie został określony przez DIM, BASIC nadaje rozmiar 10)
M.5	ELSE	IF warunek THEN instrukcja ELSE instrukcja Instrukcja lub instrukcje po THEN są wykonywane, jeżeli badany warunek jest spełniony Instrukcja lub instrukcje po ELSE są wykonywane, jeżeli badany warunek nie jest spełniony
	END	Oznacza koniec programu
	EXP(X)	Wyznacza exponent X
M5.	FIX(2.7) FIX(−2.7)	Wyznacza część całkowitą liczby PRINT FIX(2.7) → 2 PRINT FIX(−2.7) → −2
APPLE	FLASH	Powoduje okresowe wygaszanie ekranu (migotanie)

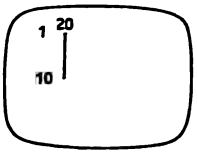
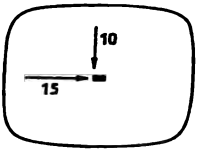
	FOR I = 1 TO 5 . . . NEXT I	Wszystkie instrukcje między FOR a NEXT są wykonywane 5 razy 10 FOR I = 1 TO 5 20 PRINT I,I*I 30 NEXT I 1 1 2 4 . . 5 25
	FOR I = 3 TO 1 STEP −1 . . . NEXT I	10 FOR I = 3 TO 1 STEP −1 20 PRINT I,I*I 30 NEXT I 3 9 2 4 1 1
	FRE(0)	Wskazuje adres wolnego obszaru w pamięci operacyjnej PRINT FRE(0)→ 15234
	FRE(X$)	Wskazuje adres wolnego obszaru dla tekstów PRINT FRE(X$) → 900
APPLE	GET X$	Czyta znak wprowadzany z klawiatury (nie wyświetlając go) 10 GET X$ 20 PRINT X$ 30 GOTO 10
	GOSUB 1000	Powoduje przeskok do instrukcji 1000 (tak jak GOTO 1000), ale po napotkaniu instrukcji RETURN wykonywanie programu jest kontynuowane od linii 110. 100 GOSUB 1000 → 1000 podprogram 110 ... ↖ 1010 ... 200 GOSUB 1000 ↖ 1020 ... 210 ... ↖ 1030 RETURN
	GOTO 1000	Powoduje przeskok do linii 1000. Instrukcji tej można użyć w 'trybie bezpośrednim' po przerwaniu programu: zmienne nie są wówczas zerowane (w odróżnieniu od RUN 1000)
APPLE	GR	Powoduje przejście do trybu graficznego niskiego poziomu
APPLE	HCOLOR = kolor	Ustala kolor punktów (0−7)
APPLE	HGR	Powoduje przejście do trybu graficznego wysokiego poziomu

APPLE	HLIN 10,20 AT 5	W trybie graficznym niskiego poziomu, wykreśla linię poziomą od kolumny 10 do kolumny 20 w wierszu 5
APPLE	HPLOT 10, 20	Zapala punkt 10, 20
	HPLOT 10, 20 TO 20, 40 TO 40, 25	Kreśli dwie linie 
APPLE	HOME	Kasuje ekran
APPLE	HTAB 10	Przesuwa kursor w poziomie do kolumny 10
		10 HTAB 10:VTAB 5 20 PRINT "XXXX" 
	IF warunek THEN instrukcja	Bada warunek i wykonuje instrukcję lub instrukcje po THEN, jeżeli warunek jest spełniony 10 INPUT "Ile masz lat?"; WIEK 20 IF (WIEK > 18) THEN PRINT "Jesteś pełnoletni"
M5. TRS80	INKEY$	Nieustannie czyta klawiaturę zwracając łańcuch pusty, jeżeli nie został wprowadzony żaden znak 10 X$ = INKEY$:IF X$ = "" THEN 10 20 PRINT X$; 30 GOTO 10
	INPUT "Rok?"; ROK	Wyświetla tekst 'Rok?' i czeka aż operator wprowadzi wartość ROK
	INPUT "Nazwisko, Wiek?"; N$, WIEK	Wyświetla tekst 'Nazwisko, Wiek?' i czeka aż operator wprowadzi wartości N$ i WIEK (rozdzielone przecinkiem). Wprowadzany tekst nie może zawierać przecinka
M5. TRS80	INSTR ("JEZYK BASIC", "BASIC")	Określa pozycję tekstu 'BASIC' w tekście 'JĘZYK BASIC', tzn. 7 10 X$ = "BASIC" 20 P = INSTR("JEZYK BASIC", X$) 30 PRINT P → 7
	INT(12.7)	Wyznacza część całkowitą liczby 12.7 PRINT INT(12.7) → 12 PRINT INT(−11.3) → −12

APPLE	INVERSE	Zmienia postać znaków wyświetlanych instrukcją PRINT na "czarne-na-białym" (patrz NORMAL)
	LEFT$("BASIC", 3)	Wydziela znaki z lewej strony tekstu PRINT LEFT$("BASIC", 3) → BAS
	LIST LIST 10 LIST 10-30	Wyprowadza cały tekst programu Wyprowadza linię 10 Wyprowadza linie o numerach od 10 do 30
M5. TRS80.	LLIST	Wyprowadza program na drukarkę
M5. TRS80	LOAD "TOTO"	Ładuje program 'TOTO' do pamięci operacyjnej
APPLE	LOAD TOTO	Ładuje program 'TOTO'
M5. TRS80	LPRINT "Dzien dobry"	Wyprowadza tekst 'Dzien dobry' na drukarkę
	MID$(X$, 7, 3)	Wydziela znaki z wnętrza tekstu 10 X$ = "BASIC DLA WSZYSTKICH" 20 PRINT MID$(X$, 7, 3) → DLA
M5. TRS80	MID$(X$, 3, 2) = "XX"	Wymienia znaki wewnątrz tekstu 5 X$ = "BBBBBB" 10 MID$(X$, 3, 2) = "XX" 20 PRINT X$ → BBXXBB
	13 MOD 4	Wyznacza resztę z dzielenia PRINT 13 MOD 4 → 1
	NEW	Kasuje program umieszczony w pamięci komputera
APPLE	NORMAL	Powrót do wyświetlania w normalnym trybie
APPLE	NOTRACE	Anuluje działanie instrukcji TRACE. Wyprowadzanie numerów wykonywanych instrukcji zostaje przerwane
	ON B GOSUB 100, 200, 500	Zależnie od wartości B (1, 2, 3), następuje przeskok do linii 100, 200 lub 500 Po powrocie z podprogramu, program jest wykonywany dalej od instrukcji następującej po ON B GOSUB... 10 INPUT "Co wybierasz? (1, 2, 3)"; WB 20 ON WB GOSUB 100, 200, 500 30 END 100 PRINT "Wariant 1": RETURN 110: 200 PRINT "Wariant 2": RETURN 210: 500 PRINT "Wariant 3": RETURN

	ON B GOTO 100, 200, 500	Tak samo jak ON B GOSUB..., ale bez możliwości powrotu
	OR	Umożliwia badanie wielu warunków. Jeżeli warunek 1 LUB warunek 2 jest spełniony, to wykonywane są instrukcje następujące po THEN 90 INPUT "X?";X 100 IF (X = 1) OR (X = 3) THEN PRINT "X jest równe 1 lub 3"
	PEEK(1000)	Podaje zawartość komórki pamięci o adresie 1000 10 X = PEEK(1000)
TRS80	POINT(X,Y)	Przyjmuje wartość –1, gdy punkt X,Y jest zapalony, a 0, gdy jest zgaszony
	POKE 1000, 12	Umieszcza wartość 12 (dziesiętnie) w komórce pamięci o adresie 1000
	PRINT PRINT "DZIEN DOBRY" PRINT "KOWALSKI": TAB (11); "JAN" PRINT USING " ###.## "; 23.546	Powoduje przeskok do nowej linii ekranu. Wyświetla 'DZIEN DOBRY' Wyświetla 'KOWALSKI', a następnie 'JAN' w kolumnie 11 Wyświetla 23.55 z dosunięciem do prawej strony (tylko dla Microsoft 5. i TRS80)
	READ MIESS. ROK	patrz DATA
	REM pleple	Wszystko to, co występuje w linii po słowie REM, jest traktowane jako komentarz W Microsoft 5. i TRS80 zamiast REM można używać znaku ' (apostrof)
M5.	RENUM RENUM 100,5	Przenumerowanie całego programu ze skokiem 10 Przenumerowanie programu od linii 100, ze skokiem 5
	RESTORE RESTORE 100	Ustawia wskaźnik danych DATA na początek, co umożliwia ponowne odczytanie zestawu danych Ustawia wskaźnik na linii DATA o numerze 100
	RETURN	Oznacza koniec podprogramu. Powoduje powrót do programu wołającego (do instrukcji następującej po GOSUB)
	RIGHT$(X$,3)	Wydziela znaki z prawej strony tekstu 10 X$ = "BASIC" 20 PRINT RIGHT$(X$,3) → SIC
TRS80	RND(100) RND(0)	Generuje całkowitą liczbę losowa z przedziału 1–100 Generuje liczbę losową z przedziału 0–1

APPLE M5.	RND(1) RND(−3) RND(0)	Generuje liczbę losową z przedziału 0−1 Inicjuje ciąg liczb losowych Podaje ostatnio wygenerowaną liczbę losową
	RUN RUN 100	Powoduje wykonanie programu Nadaje zmiennym wartość początkową 0 Powoduje wykonanie programu, poczynając od linii 100
M5. TRS80	SAVE "MULT"	Przechowuje program pod nazwą MULT
APPLE	SAVE MULT	Przechowuje program pod nazwą MULT
APPLE	SCRN(X,Y)	Podaje kolor punktu X,Y
	SGN(X)	Podaje znak liczby: 1 liczba dodatnia 0 zero −1 liczba ujemna
	SIN(X)	Wyznacza sinus X wyrażony w radianach
	SPC(3)	Wyprowadza 3 spacje 10 PRINT "XXX"; SPC(3); "XXX" → XXX XXX
	SQR(X)	Wyznacza pierwiastek kwadratowy z X (X musi mieć wartość dodatnią)
	STOP	Przerywa wykonywanie programu, który można wznowić przez CONT
	STR$(123)	Przekształca liczbę w ciąg znaków: 10 X = 123:X$ = STR$(X) 20 PRINT RIGHT$(X$, 1) → 1
M5. TRS80	STRING$(10, "*")	Generuje tekst składający się z dziesięciu znaków 10 X$ = STRING$ (10, "*") 20 PRINT X$ → **********
	TAB(15)	Ustawia kursor w piętnastej kolumnie bieżącej linii PRINT "KORWIN"; TAB(15); "Piotr" → KORWIN Piotr
APPLE	TEXT	Anuluje tryb graficzny
APPLE	TRACE	Powoduje wyprowadzanie numerów wykonywanych instrukcji (anulacja przez NOTRACE)
M5. TRS80	TRON	Powoduje wyprowadzanie numerów wykonywanych instrukcji (anulacja przez TROFF)

	VAL(X$)	Wyznacza wartość numeryczną tekstu: 10 X$ = "123 zł." 20 PRINT VAL(X$) → 123
APPLE	VLIN 1,10 AT 20	W grafice niskiego poziomu kreśli linię pionową od wiersza 1 do wiersza 10 w kolumnie 20 
APPLE	VTAB 10	Przesuwa kursor pionowo do wiersza 10 10 VTAB 10: HTAB 15 

Dodatek III

Sygnalizacja błędów

Jeżeli wysłany przez Basic komunikat nie wystarcza do zlokalizowania błędu, spróbuj **wyprowadzić w trybie bezpośrednim** wartości zmiennych (przed zmodyfikowaniem programu, ponieważ modyfikacja powoduje wyzerowanie zmiennych).

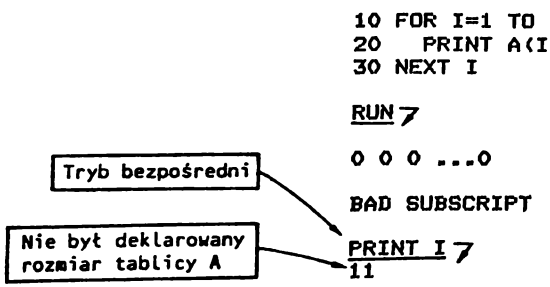
BAD SUBSCRIPT (błędny indeks)

Próba odwołania do elementu poza tablicą.

```
10 DIM A(13)
   :
70 PRINT A(15)
```

Należy zadeklarować tablicę piętnastoelementową.

Błąd ten może wystąpić także wówczas, gdy rozmiar tablicy nie został zadeklarowany instrukcją DIM (Basic przyjmuje więc, że tablica ma 10 elementów). W takim przypadku PRINT A (11) spowoduje wystąpienie błędu.



CAN'T CONTINUE (nie można kontynuować)

Nie można wznowić programu:
Instrukcja została zmodyfikowana.

DIVISION BY ZERO (dzielenie przez zero)

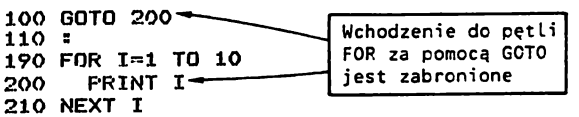
Próba dzielenia przez zero (dołącz do programu badanie IF X = 0 THEN ...).

ILLEGAL DIRECT (błąd trybu bezpośredniego)

Wprowadzonej instrukcji nie można używać w trybie bezpośrednim (np. INPUT).

NEXT WITHOUT FOR (NEXT bez FOR)

Napotkana została instrukcja NEXT, podczas gdy nie była uprzednio wykonywana instrukcja FOR.



Powinno być: 100 GOTO 190.

OUT OF DATA (brak danych DATA)

Próba odczytania danej DATA instrukcją READ, podczas gdy wszystkie dane DATA zostały już odczytane.

OUT OF MEMORY (brak miejsca w pamięci)

OVERFLOW (nadmiar)

Wartość liczby przekracza dopuszczalny zakres.

REDIMENSIONNED ARRAY (powtórna deklaracja rozmiaru tablicy)

Próba zadeklarowania rozmiaru tablicy, której rozmiar był już deklarowany. Wystąpiło już odwołanie do elementu tablicy, co spowodowało nadanie jej przez Basic standardowego rozmiaru 10, a następnie próbuje się zadeklarować rozmiar tablicy w sposób jawny.

```
10 A(5)=123
20 DIM A(15)

RUN ↵

REDIMENSIONNED ARRAY
```

RETURN WITHOUT GOSUB (powrót bez wywołania)

Napotkana została instrukcja RETURN, podczas gdy nie była uprzednio wykonywana instrukcja GOSUB.

```
 10 GOSUB 100
 30 :
100 PRINT "podprogram"
110 RETURN
```

Zapomniano o wstawieniu instrukcji END lub STOP za linią 10: podprogram został wykonany dwukrotnie.
(Należy wstawić: 20 STOP).

STRING TOO LONG (za długi tekst)

Długość tekstu przekroczyła 255 znaków.

```
10 X$=X$+"A"
20 GOTO 10
```

SYNTAX ERROR (błąd składniowy)

TYPE MISMATCH (niezgodność typów)

Próba przyporządkowania wartości numerycznej zmiennej tekstowej (lub odwrotnie).

```
10 X$=123          ( powinno byc X$="123"  lub  X=123 )
10 X="AZERTY"      ( powinno byc X$="AZERTY" )
```

UNDEF'D STATEMENT (nieokreślony numer linii)

Instrukcja GOTO lub GOSUB odwołuje się do nieistniejącej linii.

UNDEF'D FUNCTION (brak definicji funkcji)

Próba wywołania funkcji, która nie została zdefiniowana.

REDO FROM START (zacznij od początku)

Próba wprowadzenia tekstu w odpowiedzi na instrukcję INPUT, podczas gdy oczekiwana jest wartość numeryczna.

```
10 INPUT "Liczba? ";X
```

INPUT oczekuje wartości numerycznej

```
RUN ↵

Liczba? KOWALSKI ↵
REDO FROM START? 1234 ↵
```

EXTRA IGNORED (nadmiarowe dane)

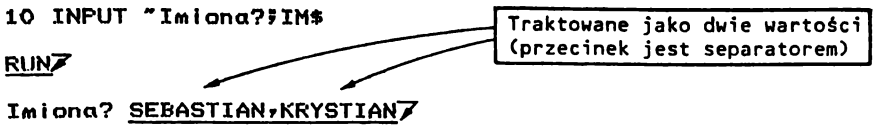
Próba wprowadzenia większej, niż zostało to przewidziane, liczby danych.

```
10 INPUT "Miesiac, Rok? ";MIES,ROK
```

INPUT oczekuje jedynie podania dwóch wartości

```
RUN ↵

Miesiac, Rok? 13,12,45 ↵
```

45 jest ignorowane.



Przyjmowany jest jedynie tekst SEBASTIAN (nie należy wprowadzać przecinka, ewentualnie, w systemie Microsoft 5., trzeba użyć konstrukcji LINE INPUT).

Dodatek IV

Tablica kodu ASCII

0	@ c	32	SPACJA	64	@	96	`
1	Ac	33	!	65	A	97	a
2	Bc	34	"	66	B	98	b
3	Cc	35	#	67	C	99	c
4	Dc	36	$	68	D	100	d
5	Ec	37	%	69	E	101	e
6	Fc	38	&	70	F	102	f
7	Gc (BELL)	39	'	71	G	103	g
8	Hc (BS)	40	(	72	H	104	h
9	Ic (HT)	41	)	73	I	105	i
10	Jc (LF)	42	*	74	J	106	j
11	Kc	43	+	75	K	107	k
12	Lc	44	,	76	L	108	l
13	Mc (CR)	45	−	77	M	109	m
14	Nc	46	.	78	N	110	n
15	Oc	47	/	79	O	111	o
16	Pc	48	0	80	P	112	p
17	Qc	49	1	81	Q	113	q
18	Rc	50	2	82	R	114	r
19	Sc	51	3	83	S	115	s
20	Tc	52	4	84	T	116	t
21	Uc	53	5	85	U	117	u
22	Vc	54	6	86	V	118	v
23	Wc	55	7	87	W	119	w
24	Xc	56	8	88	X	120	x
25	Yc	57	9	89	Y	121	y
26	Zc	58	:	90	Z	122	z
27	[c (ESC)	59	;	91	[	123	{
28	\c	60	<	92	\	124	|
29	]c	61	=	93	]	125	}
30	^c	62	>	94	^	126	~
31	-c	63	?	95	_	127	DEL

Uwaga: Ac oznacza Ctrl A.

WNT Warszawa 1987.
Wydanie I. Nakład 80.000 + 300 egz.
Ark. wyd. 8,1. Ark. druk. 11,0.
Format B-5. Papier offset kl. III 65 g. rola 70 cm
Pracę oddano do składania w czerwcu 1986 r.
Podpisano do druku w czerwcu 1987 r.
Druk ukończono w czerwcu 1987 r.
Symbol Ek 82179·WNT.
Drukarnia im. Rew. Październikowej
Warszawa
Zam. 2574 11 86. K-13.

Cena zł 380.—

Wprowadzenie do języka Basic

Naucz się programowania:
Z książką „Basic dla wszystkich" w ręku,
usiądź przed komputerem i zacznij wprowadzać instrukcje.
Bardzo szybko przyswoisz sobie podstawowe pojęcia z zakresu programowania:
zmienne, decyzje, pętle...
Będziesz już „rozumiał" komputer
i wiedział, czego można się po nim spodziew
Ale to jeszcze nie wszystko...
Dzięki licznym, dobrze zilustrowanym przy
i na bieżąco objaśnianym programom,
zdobędziesz bez żadnych trudności
niezbędne podstawy, które umożliwią Ci
pogłębianie wiadomości i pisanie własnych
(gry, programy z dziedziny zarządzania, nauczan

-204-0858-X